

T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BSM 498 BİTİRME ÇALIŞMASI

**KORONAVİRÜSÜN DERİN ÖĞRENME TEKNİKLERİ
İLE TESPİT EDİLMESİ**

G121210053 - ŞÜKRÜ UFUK AKSOY

B130910044 - GÖKHAN SIBIÇ

B161210057 - EMRE ÇAKMAK

Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ
Tez Danışmanı : DR.ÖĞR. ÜYESİ SERAP ÇAKAR

2020-2021 Bahar Dönemi

T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

**KORONAVİRÜSÜN DERİN ÖĞRENME
TEKNİKLERİ İLE TESPİT EDİLMESİ**

BSM 498 - BİTİRME ÇALIŞMASI

ŞÜKRÜ UFUK AKSOY

GÖKHAN SIBIÇ

EMRE ÇAKMAK

Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ

Bu tez .. / .. / ... tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

.....
Jüri Başkanı

.....
Üye

.....
Üye

ÖNSÖZ

Mühendislik gibi zor bir bölümü bitirebilmek için sabır, özveri ve odaklanmak gerekir. Eğitim hayatımız boyunca bizlere destek veren ailelerimize çok teşekkür ederiz.

Yapay zeka alanında takip ettiğimiz tamamen Türkçe içerikli olan ve bizlere katkısı çok büyük olan DATAI ekibine ve takıldığımız her sorunun cevabını Türkçe bir şekilde bulabileceğimiz bir forum oluşturmuş olan Global AI HUB ekibine teşekkür ederiz.

Proje çalışması doğrultusunda çalışmamızda bizlere yardımcı olan danışman hocamız Dr.Öğr.Üyesi SERAP ÇAKAR'a teşekkürü borç biliriz.

İÇİNDEKİLER

ÖNSÖZ.....	iv
İÇİNDEKİLER.....	v
ŞEKİLLER LİSTESİ.....	vii
ÖZET.....	ix
BÖLÜM 1.	
GİRİŞ.....	1
BÖLÜM 2.	
VERİ SETİNİN ELDE EDİLMESİ.....	3
2.1 Veri Setinin Elde Edilme Yöntemleri.....	3
2.1.1 Ham Verilerden Elde Edilen Veri Setleri.....	4
2.1.1.1 Verilerin Toplanması.....	4
2.1.1.2 Verilerin Etiketlenmesi ve Sınıflandırılması.....	5
2.1.2 Hazır Veri Setleri.....	5
2.2 Modelde Kullanılacak Veri Setinin İncelenmesi.....	6
2.2.1 Veri Seti Yapısı.....	7
2.2.1.1 Eğitim Veri Seti (Train).....	9
2.2.1.2 Doğrulama Veri Seti (Valid).....	9
2.2.1.3 Test Veri Seti (Test).....	9
BÖLÜM 3.	
DERİN ÖĞRENME VE EVRİŞİMSEL SİNİR AĞLARI.....	10
3.1 Yapay Sinir Ağları.....	10
3.1.1 Maliyet Fonksiyonu.....	12
3.1.2 İleri Yönde Yayılım.....	13
3.1.3 Geri Yönde Yayılım.....	14
3.1.4 Hiper Parametreler.....	16
3.1.4.1 Öğrenme Katsayısı.....	17
3.1.4.2 Aktivasyon Fonksiyonu.....	17
3.1.4.2.1 Doğrultulmuş Lineer Birim (ReLU) Aktivasyon	18
Fonksiyonu.....	
3.1.4.2.2 Aktivasyon Fonksiyonu Seçim.....	19
3.1.4.3 Eniyileme Türleri.....	21
3.1.4.4 Epok, Döngü Sayısı ve Paket Boyutu.....	21
3.2 Derin Öğrenme ve Evrişimsel Sinir Ağları –CNN.....	22
3.2.1 Derin Öğrenme – Deep Learning.....	22
3.2.2 Evrişimli Sinir Ağları (CNN).....	24
3.2.2.1 Artık Sinir Ağı (ResNet).....	25
3.2.2.1.1 ResNet18 ve ImageNet.....	27
3.2.3 Evrişimli Sinir Ağları Mimarisi.....	27
3.2.3.1 Evrişim Katmanı.....	28
3.2.3.2 Havuzlama/Ortaklaşma Katmanı.....	31

3.2.3.3 Tam Baęlaşımli Katman.....	32
3.2.4 Evriřimli Sinir Aęlarında Dolgu Ekleme ve Adım Sayısı.....	32
3.2.4.1 Dolgu/Piksel Ekleme.....	32
3.2.4.2 Adım Sayısı.....	33
3.2.5 PyTorch ve Tensörler.....	34
BÖLÜM 4.	
PROJE MODELİNİN OLUřTURULMASI.....	36
4.1 Kütüphanelerin Girilmesi ve Veri Setinin Oluřturulması.....	36
4.2 Görüntü Dönüřümleri.....	37
4.3 Veri Yükleyici (DataLoader) Hazırlama.....	38
4.4 Veri Görselleřtirme.....	39
4.5 ResNet18 ile Modelin Oluřturulması.....	40
4.6 Modelin Eęitilmesi.....	41
BÖLÜM 5.	
SONUÇLAR VE ÖNERİLER.....	42
KAYNAKLAR.....	44
ÖZGEÇMİř.....	45
BSM 498 BİTİRME ÇALIřMASI DEęERLENDİRME VE SÖZLÜ SINAV	46
TUTANAęI.....	

ŞEKİLLER LİSTESİ

Şekil 1.1.	Veri Seti Sınıflarından Örnek Görüntüler.....	2
Şekil 2.1.	“COVID-19 Radiography Dataset” adlı Veri Setinden Etiketlenmiş Görüntüler.....	7
Şekil 2.2.	Veri Seti Modeli Örnekleri (a) ve (b).....	8
Şekil 2.3.	Eğitim yapan modelinin akış diyagramı.....	8
Şekil 3.1.	Yapay sinir ağının, insan nöronunu referans alarak modellenmesi.....	11
Şekil 3.2.	Yapay sinir ağı modeli.....	11
Şekil 3.3.	Gizli katmanlara sahip yapay sinir ağı.....	13
Şekil 3.4.	Farklı değerlerdeki öğrenme katsayılarının sonuç-etki çıktıları.....	17
Şekil 3.5.	Aktivasyon fonksiyonları ve türevlenmiş halleri.....	18
Şekil 3.6.	ReLU fonksiyonu ve türevi.....	19
Şekil 3.7.	Aktivasyon fonksiyonlarının türevleri.....	20
Şekil 3.8.	Eğim düşümünün uygulanması.....	21
Şekil 3.9.	Doğruluk ve epok sayısının arasındaki ilişki.....	22
Şekil 3.10.	Makine öğrenmesi ve derin öğrenme modelleri.....	23
Şekil 3.11.	2012-2017 yılları arasında dereceye giren ağların hata-doğruluk oranları.....	24
Şekil 3.12.	ResNet yapısı, $\ell - 1$ katmanı, $\ell - 2$ 'den etkinleştirme.....	25
Şekil 3.13.	1×1 Evrişim içeren ve içermeyen ResNet bloğu.....	26
Şekil 3.14.	ResNet-18 Mimarisi.....	27
Şekil 3.15.	3×3 Filtre.....	29
Şekil 3.16.	Evrişim işlemi örneği.....	29
Şekil 3.17.	Çaprazkorelasyon fonksiyonu ile evrişim işlemi.....	30
Şekil 3.18.	Ortaklaşma işlemi gösterimi.....	31
Şekil 3.19.	Girdi matrisine dolgu/piksel ekleme örneği.....	33
Şekil 3.20.	Tensör yapısı.....	34
Şekil 4.1.	Modele girdi olarak verilen görüntülerin model tarafından tahminlenmesi.....	36

Şekil 4.2.	Verisetinin çoğaltılması için görüntülerin yatay formatta değiştirilmesi.....	38
Şekil 4.3.	Veri yükleyici genel yapısı.....	38
Şekil 4.4.	Veri setlerinden rastgele elde edilen verilerin görselleştirilmesi.....	39
Şekil 4.5.	Model eğitiminden önce ResNet ağ modelinin tahminleme çıktıları.....	41
Şekil 4.6.	Resnet18 katmanları parametreleri.....	41

ÖZET

Anahtar kelimeler: Derin Öğrenme, Yapay Sinir Ağları, CNN, COVID-19 Tespiti

Koronavirüs (COVID-19), bu tezin yazıldığı süreçte (2021), tüm dünyaya kısa bir sürede yayılarak milyonlarca insanın ölümüne, vücutlarda kalıcı hasarlar bırakan ve dünya genelinde bir pandeminin ilan edilmesi sebep olmuş bir virüs türüdür.

Yakalanıldığında hayati tehlike oluşturan koronavirüsünün tespiti için tanı setlerinin yetersiz olması, uygulanan testlerdeki yanılma paylarının da göz önünde bulundurulması sonucunda makine öğrenmesi ve derin öğrenme gibi günümüzde her alanda insanlığa kolaylıklar sağlayan bu teknolojiler de kullanılmaya başlanılmış ve bu alanda akademik çalışmalar da salgının yayılması ile birlikte artmıştır. Bu destekleyici teknolojiler sayesinde, hastalığın erken teşhisinde kullanılan radyolojik görüntüler süreç boyunca iş yükü artmış olan sağlık çalışanlarına da büyük kolaylıklar sağlamaktadır.

Bitirme tezinin çalışma konusu olan koronavirüsünün tespiti için akciğer tomografi görüntüleri ve akciğer röntgen görüntüleri üzerinden veri setleri oluşturularak model üzerinde kullanılmıştır. Veri setinin araştırılması sonucunda açık kaynak olarak Kaggle platformunda bulunan “COVID-19 Radiography Dataset” adlı veri seti kullanılmıştır.

Bu çalışmada toplamda 43 sınıftan oluşan ve toplam 39.209 eğitim, 12.630 test ve 4410 doğrulama (valid) görüntüsü içeren German Traffic Sign Recognition Benchmark (GTSRB) veri seti kullanılmıştır. Derin öğrenme yöntemlerinden Konvolüsyonel Sinir Ağları (CNN) kullanılarak sınıflandırma işlemi gerçekleştirilmiştir.

Python Programlama dili ile PyTorch derin öğrenme kütüphanesi kullanılarak oluşturulan ResNet18 sınıflandırma modeli ile sınıflandırma işlemi sonucunda ortalama %94 oranında başarı elde edilmiştir. Test doğruluk oranı ise ortalama %93 olarak elde edilmiştir.

BÖLÜM 1. GİRİŞ

Koronavirüs (COVID-19), tüm dünyada çok kısa sürede yayılan ve ölümcül sonuçları olan bir salgın hastalıktır. Bu tür bulaşıcı hastalıkların insanlara zarar vermeden veya minimum zararlar doğru bir şekilde tespit edilmesi ve gerekli tedavinin erken süreçte başlatılması gerekmektedir [1].

Uzun süredir yeryüzünde bulunan bu virüs türünün yeni tipi ilk olarak Çin'in Wuhan bölgesinde, 2019 Aralık ayının başında görülmüş olup, 30 Ocak 2020 tarihinde Dünya Sağlık Örgütü (WHO) tarafından küresel bir sağlık acil durumu ilan edilmiştir. 11 Mart 2020 tarihinde ise virüs pandemi, yani küresel bir salgın hastalık olarak ilan edilmiştir. Bu bitirme tezinin hazırlandığı sürede toplamda yaklaşık 131.5 milyon insan virüse yakalanmış olup, 105.9 milyonu hastalığı atlatabilmişken, 2.8 milyon insan da hayatını kaybetmiştir [1].

Dünya genelinde pandemi olarak ilan edilen ve insanlarda hayati tehlike oluşturan koronavirüsünün tespiti için tanı setlerinin yetersiz olması, uygulanan testlerdeki yanlış paylarının da göz önünde bulundurulması sonucunda makine öğrenmesi ve derin öğrenme gibi günümüzde her alanda insanlığa kolaylıklar sağlayan bu teknolojiler de kullanılmaya başlanılmış ve bu alanda akademik çalışmalar da salgının yayılması ile birlikte artmıştır [2]. Bu destekleyici teknolojiler sayesinde, hastalığın erken teşhisinde kullanılan radyolojik görüntüler süreç boyunca iş yükü artmış olan sağlık çalışanlarına da büyük kolaylıklar sağlamaktadır.

Son zamanlarda koronavirüsün tespiti için akciğer radyolojik görüntüleri üzerinden yapay zeka teknolojileri kullanılmaktadır. Radyolojik görüntüler üzerinde Vgg16, InceptionV3, AlexNet, GoogLeNet, ResNet50, SqueezeNet derin öğrenme metotlarını kullanarak virüs içeren akciğerlerin tespiti hızlı bir şekilde belirlenebilir. Kullanılan

sistemlerde çalışan algoritmalar, öğrenme modelleri ile özellik çıkarımı (feature map) yapılarak sınıflandırılmakta ve bulanık mantık, SVM (destek vektör makinesi) ve görüntü işleme teknikleri kullanılarak koronavirüs tespiti yapılmaktadır [3] [4].



(a)

(b)

(c)

Şekil 1.1. Veri Seti Sınıflarından Örnek Görüntüler

(a) Sağlıklı birey akciğer görüntüsü, (b) COVID-19 pozitif olan birey akciğeri, (c) Zatürre hastalığı olan birey akciğeri

Literatür taraması sonucunda elde edilen bilgilere dayanarak, çalışmalara başlanılan ve bitirme tezinin çalışma konusu olan koronavirüsün tespiti için akciğer tomografi görüntüleri ve akciğer röntgen görüntüleri üzerinden veri setleri oluşturularak model üzerinde kullanılmıştır [3] [4] [5]. Veri setinin araştırılması sonucunda veri setlerinin ücretsiz bir şekilde paylaşıldığı platform olan Kaggle’da bulunan “COVID-19 Radiography Dataset” adlı veri seti kullanılmaktadır. Veri seti ile ilgili ayrıntılar Bölüm 2.’de açıklanmıştır.

BÖLÜM 2. VERİ SETİNİN ELDE EDİLMESİ

Makine öğrenmesi ve derin öğrenme temelli oluşturulan modeller, veri setlerine ihtiyaç duymaktadırlar. Modellerin öğrenmeyi gerçekleştirebilmesi için, belirli bir oranda (İhtiyaç duyulan miktar, model performansının beklentisine göre değişebilmektedir.) veri setinin modelin parametre çıkarımının sağlanması için girdi olarak verilmesi gerekmektedir. Bu doğrultuda modelin, veri setinden özellikler çıkararak istenilen şekilde çalışması öngörülmekte olup, modelden beklentilere göre veri setinin oranlarında artırma veya azaltma ihtiyaçları söz konusu olabilmektedir.

İhtiyaç duyulan veri setleri birkaç şekilde elde edilebilme söz konusudur. Veri seti elde etme yöntemleri ve model üzerinde kullanılmış olan veri setinin özelliklerinden bu bölümde bahsedilecektir.

Projede kullanılan veri seti, derin öğrenme modeli üzerinde fotoğraf temelinde eğitileceği için, anlatılmakta olan veri seti bölümü de fotoğraf temelli veri setleri üzerinde yoğunlaşmaktadır.

2.1 Veri Setinin Elde Edilme Yöntemleri

MModelin çıktıları oluşturabilmesi için öncelikle modellerde bulunan sınıflardaki farklılıklardan yola çıkarak özellik haritası çıkarması gerekmektedir. Bu doğrultuda girdi olarak sunulan veri setleri model öğrenmesinde en önemli faktörler arasında yer almaktadır.

Veri setleri, oluşturulan modellerin kullanılma amaçlarına göre farklılıklar içerebilmektedir. Bu doğrultuda, modelden istenilen çıktılara göre, hazır hale getirilmiş durumda olan veri setlerinin kullanımı veya sadece model özelinde kullanılması amaçlanan ve sıfırdan oluşturulan veri setlerinin kullanımı olmak üzere farklılıklar söz konusu olabilmektedir.

2.1.1 Ham Verilerden Elde Edilen Veri Setleri

Oluşturulması planlanan modelin, daha özel bir amaç doğrultusunda kullanılması, açık kaynak olarak bulunan veya satın alınabilen veri setlerinin model ile uyumsuzluğu, eldeki veri setlerinin yetersiz olması durumu vb. veri setinin sıfırdan oluşturulması veya takviyesi durumları söz konusudur. Bu bağlamda veri setleri ham veri olarak elde edilmektedir. Elde edilen ham verilerden veri setinin oluşturabilinmesi için, model koşullarına uygun olacak şekilde seçilmesi ve sınıflarına ayrılacak şekilde etiketlenmesi gerekmektedir.

Veri setinin, modelde kullanılabilecek hale gelebilmesi için belirli aşamalar üzerinden geçmesi gerekmektedir.

2.1.1.1 Verilerin Toplanması

Oluşturulan ve/veya oluşturulmuş model, fotoğraf temelli bir model üzerinde durulduğu için, verilerin elde edilmesi fotoğraflar üzerinden ilerlemektedir. Öncelikle ham veri (raw data) elde edilmesi gerekmektedir. Burada izlenicelek yöntemler, modeli üreten kişiler internet üzerinden beautifulsoup gibi Python kütüphanesi kullanarak veri çekebilirken, kendileri de istenilen veri setini fotoğraf veya video çekme aracılığıyla elde edebilmektedir.

Video kaydederek çekilen verilerin CNN tarafından girdi olarak sunulabilecek formata dönüştürülmesi yani videoların çerçevelere (frame) dönüştürülmesi gerekmektedir.

2.1.1.2 Verilerin Etiketlenmesi ve Sınıflandırılması

İnternet üzerinden veya fotoğraf/video ile elde edilen verilerin model tarafından tanımlanması gerekmektedir. Bu doğrultuda, modelin fotoğrafların ayırımını yaparak özellikleri çıkarması için verilerin etiketlenmesi gerekmektedir.

Etiketleme işlemleri farklı şekillerde yapılarak modele girdi olarak verilebilmektedir. Kullanılan model, bir fotoğraf üzerinde belirli bir alanın tanımlanmasını ve öğrenmesini gerektiriyorsa, modele fotoğraf girdi olarak verilir. Model fotoğrafının tamamını parametre olarak alır ancak özellik haritasında sınıflandırmasını kolaylaştıracak farklılıkları etiketlenmiş olan alanın üzerinden öğrenmektedir. Bu doğrultuda, LabelImg gibi fotoğrafların etiketlenebilmesini sağlayan programlar bulunmaktadır. Program JSON formatında çıktı verirken, fotoğrafın -dörtgen formatı baz alınarak- çıktısını vermektedir. Elde edilen JSON formatı katman numarası (görüntünün sınıflandırıldığı ismin indeks numarası- ve dörtgenin koordinat bilgileri dönüşümler sonucunda (labelNo, x1, y1, x2, y2) formatı modele .txt şeklinde "Label" isimli klasör vasıtasıyla verilmektedir. Örneğin COCO dataseti fotoğrafların belirli bölümlerinde bulunan görüntülerin etiketlenmesi ile oluşturulmuştur. Diğer bir yöntem ise fotoğrafların klasörlere paylaştırılarak sınıflandırılması ve sınıf içerisinde bulunan fotoğrafın tamamının özelliklerinin model tarafından çıkartılması beklenmektedir. Yapılan proje doğrultusunda 3 farklı sınıf bulunmaktadır ve bu 3 sınıf içerisinde son bahsedilen yöntem ile oluşturulmuş olan veri seti kullanılmaktadır.

2.1.2 Hazır Veri Setleri

Verinin çok önemli olduğu ve verilerden anlam çıkarmanın günümüzde çok popüler olduğu bir dönemde bazı kurum/kuruluşlar ve gönüllü kişiler tarafından açık

kaynak kodlamayı destekleyecek şekilde veri setlerini insanlara sunmaktadırlar. Bu doğrultuda Kaggle gibi veri setlerini sunan ve yarışmalar düzenleyen bir platformlardan da veri setleri bulunmaktadır. COCO gibi binlerce objenin etiketli halde bulunduğu açık kaynak olarak sunulan veri setleri modeller de bulunmaktadır.

Proje kapsamında da hazır olarak sınıflarına ayrıştırılmış olan COVID-19 Radiography adlı veri seti kullanılmaktadır. Bu veri seti hakkında detaylı bilgiler Bölüm 2.2’de bulunmaktadır.

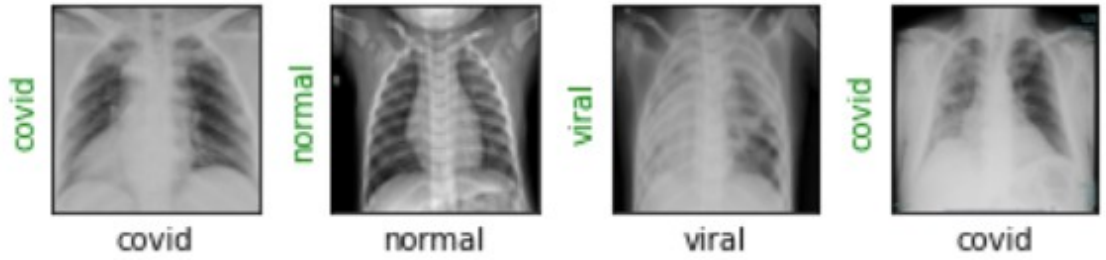
2.2 Modelde Kullanılacak Veri Setinin İncelenmesi

Veri setlerinin ham şekilde elde edilmesi, ayıklanması ve etiketlenmesi işlemleri çok uzun sürdüğü ve model için elde edilmesi gereken veri setinin kişisel hukiki süreçler doğrultusunda toplanma süreci uzun sürmektedir. Ancak bu tez yazılırken bulunduğumuz pandemiden dolayı tüm dünya halihazırda zaten COVID-19 ile mücadele vermektedir. Bölüm 1’de de bahsedildiği gibi bu alanda derin öğrenme ve makine öğrenmesi ile çalışmalar yapılarak sürecin kolaylaştırılması amaçlanmaktadır. Bu yüzden COVID-19 üzerine yapılan çalışmaların yanı sıra herkesin erişebilmesi için Kaggle gibi platformlarda veri setleri paylaşılmaktadır.

Modelin ihtiyacı duyduğu ve projede kullanılan veri seti, Kaggle platformu üzerinde açık kaynak olarak eğitim amaçlı sunulan COVID-19 Radiography Dataset adlı veri seti proje üzerinde kullanılmaktadır. Bu veri setinde toplamda 3000 adet olmak üzere akciğer grafileri yani röntgenleri bulunmaktadır. Veri seti eğitilecek ve test edilecek verileri ikiye ayırmış halde bulundurmaktadır. Her iki veri setinde de üç farklı sınıflandırma yapılmaktadır. Bunlar; Normal, Covid ve Viral olmak üzere sınıflandırılmıştır.

Eğitilecek veri seti içerisinde toplamda 2815 görüntü bulunmaktadır. Bunlardan “Normal” olarak adlandırılan sınıf içerisinde toplamda 1311 adet normal akciğer röntgen görüntüleri bulunmaktadır. Viral olarak adlandırılan ve hastalıklı olan diğer akciğer röntgen görüntüleri 1315 adet iken, Covid adlı sınıf içerisinde bulunan akciğer röntgen sayısı ise 189 adettir.

Test veri seti içerisinde ise 90 adet görüntü bulunmaktadır. Test veri seti içerisinde de 3 başlık altında sınıflandırılan bu görüntülerin her bir sınıfa 30’ar adet görüntü düşmektedir.



Şekil 2.1 “COVID-19 Radiography Dataset” adlı Veri Setinden Etiketlenmiş Görüntüler

Veri setlerinin .csv formatta veya direkt olarak .jpg formatta olmalarının yanı sıra Python Programlama dilinin sunduğu büyük bir avantaj olan Pickle adlı kütüphane ile veri setlerinin formatları aynı kalırken boyutları matrislerle tutulduğu için daha düşük ve kolay taşınabilir olmaktadır. Bu doğrultuda eğitim, doğrulama ve test veri setleri “.p” formatına çevrilerek kullanılmıştır.

2.2.1 Veri Seti Yapısı

Bu bölümde, bir CNN modeli için veri setinin nasıl yapılanması gerektiğinden, proje kapsamında kullanılan ResNet18 modeline verilen veri setinden ve veri setlerindeki eğitim, test, doğrulama ayrıştırılmalarından bahsedilmektedir.

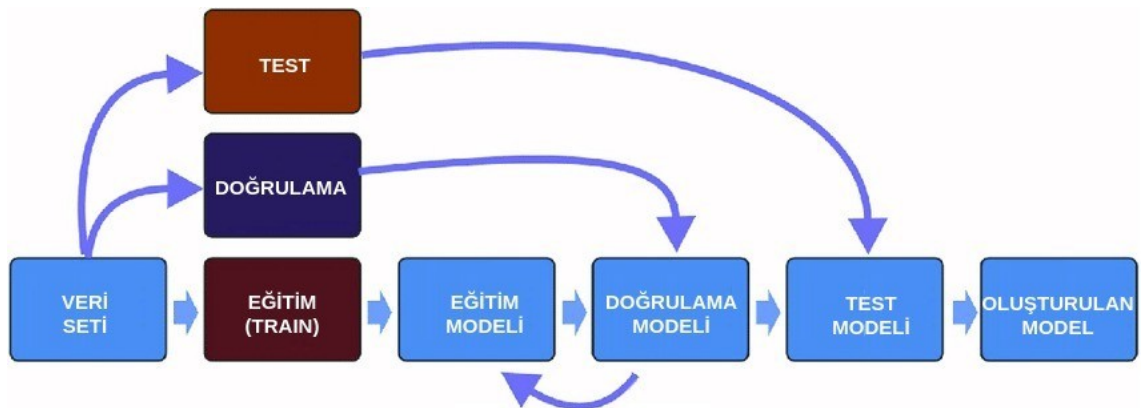
Veri setlerinin düzenlenmesinde modeller üzerinde optimum sonuçların alınması için veri setlerinin %80'i eğitim; %20'lik kısmı da test için kullanılır. Bazen veriler %60 eğitim, %20 doğrulama (validation), %20 test olarak da ayrılmaktadır. Şekil 2.2'te şematize olarak gösterilmektedir.

Projedeki bulunan hazır veri setinde yaklaşık %85'lik bir eğitim veri seti oluşturulurken, %15'lik kısmı da test olarak ayrılmıştır. Oluşturulan model de bu veriler doğrultusunda verim elde etmektedir.



Şekil 2.2 Veri Seti Modeli Örnekleri

Bir model eğitilirken eğitim, test, doğrulama veri setleri ile çıktı oluşturmaktadır. Şekil 2.3'te bu 3 farklı veri seti ile oluşturulan model çıktısı verilmektedir.



Şekil 2.3. Eğitim yapan modelin akış diyagramı

2.2.1.1 Eğitim Veri Seti (Train)

Modelin öğrenmesi için modele sunulan veri seti bu klasör üzerinden gelmektedir. Model üzerinde gradyan (Eğim-Gradient) hesaplaması ve ağırlık (weight) güncellemesi işlemleri yapılır. Train klasörü altındaki tüm girdiler matris formunda (X-matrisi) girilerek ağırlıkları (w - weight matrisi), bias (b matrisi) ile işleme tabii tutularak herbir katman sonucunda loss function denilen kayıp fonksiyonu hesaplanarak bir sonraki katmanda o değer, girdi alınarak en son katmana kadar ilerler. Türevleme işlemleri ile ileri doğru (forward propagation) ve de geri doğru yayılım (backward propagation) yöntemleri ile ağırlık optimize bir şekilde öğrenim gerçekleştirdiği kısımdır.

2.2.1.2 Doğrulama Veri Seti (Valid)

Eğitim ilerledikçe eğitim kalitesini değerlendirmek için gerçekleştirilen çapraz doğrulama (Cross-Validation) için kullanılmaktadır.

Çapraz Doğrulama (Cross-Validation), algoritmanın genelleme yeteneğini kaybetmesini sağlayan ve eğitim setinin detaylı bir şekilde sınıflanmasına odaklanan over-fitting (aşırı uyma) probleminden kaçınmak için kullanılmaktadır.

2.2.1.3 Test Veri Seti (Test)

Eğitilmiş veri setinin test edilmesi için kullanılmaktadır. Bu test işlemi sonucunda modelin incelenme durumu söz konusudur.

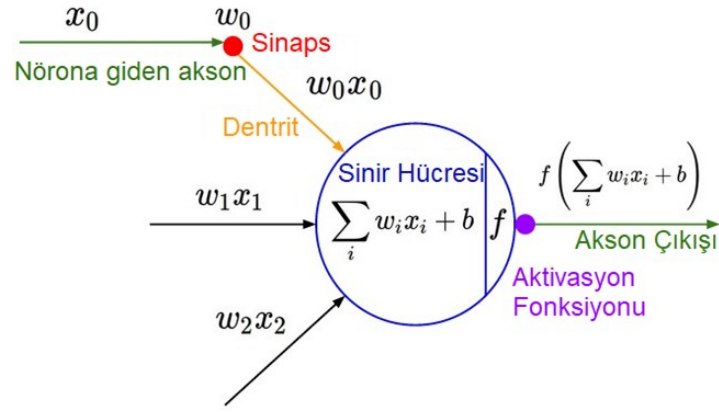
BÖLÜM 3. DERİN ÖĞRENME VE KONVOLÜSYONEL SİNİR AĞLARI (CNN – CONVOLUTIONAL NEURAL NETWORK)

Bu bölümde, elde edilmiş veri setinin, model üzerinden geçirilerek eğitilmesinden bahsedilmeden önce Derin Öğrenme (deep learning) ve Konvolüsyonel Sinir Ağlarından bahsedilecektir.

Eldeki röntgen görüntüsü veri setinin model üzerinden sınıflandırılması için kullanılan ResNet18 modeli bir derin öğrenme tekniğidir. Model içerisinde kullanılan katmanların temelinde ise evrişimli sinir ağlarının kullanıldığı ve bu sistemin açıkça anlaşılabilmesi için öncelikle yapay sinir ağlarından bahsedilecektir. Yapay sinir ağları makine öğrenmesi ve derin öğrenmede sıkça kullanılmaktadır.

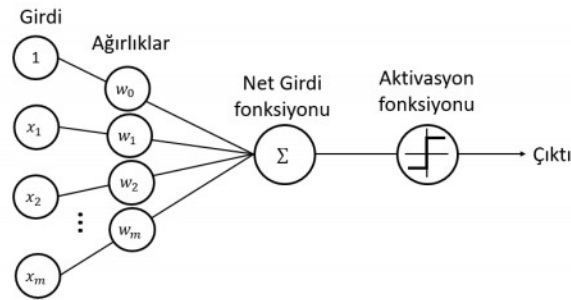
3.1 Yapay Sinir Ağları

Sinir ağları, insan beynindeki yapılan hücrelerin algılaması, öğrenmesi ve bilgileri depolaması gibi özelliklerinin taklit edilmesini amaçlayarak ortaya çıkarılmıştır. İnsan beyniyle yapılan bu işlemlerin bilgisayarlarla da yapılabilmesi öngörülmekte ve matematiksel olarak yaptırılması hedeflenmektedir. Şekil 3. 1’de insan beyinde bulunan sinir nöronun, yapay bir sinir ağdaki modellenmesi gösterilmektedir. Yapay sinir ağının referansı bu yönde oluşmaktadır.



Şekil 3.1. Yapay sinir ağının, insan nöronunu referans alarak modellemesi

Bir sinir ağı giriş, hesaplama ve çıkış katmanı olmak üzere üç bölümden oluşmaktadır. Giriş katmanındaki girdi değerleri matris halinde; x_1, x_2, \dots, x_n ve ağırlıkları da yine matris halinde; w_1, w_2, \dots, w_n olarak gösterilmektedir. Şekil 3.2’de bir yapay sinir ağı modeli gösterilmiştir.



Şekil 3.2.Yapay sinir ağı modeli

Yapay sinir ağının ürettiği çıktı sonucu sıfır ve birlerden oluşmaktadır. Bir yapay sinir ağının girdi katmanı ile ağırlıkların çarpımına kutuplama değeri eklendiğinde, sonuç eşik değerinden büyük ise sinir ağı 1, değilse 0 olarak çıktı üretecektir. Denklem 3. 1 de çıkış üreten fonksiyon gösterilmiştir ve y çıktı, X^j girdi katmanındaki matrisi, W^j ağırlıklar matrisi, b kutuplama matrisi olarak gösterilmiştir.

$$x = \sum_j w_j x_j + b$$

$$y = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

(3.1)

3.1.1 Maliyet Fonksiyonu

Maliyet fonksiyonu, üretilen yapay sinir ağı çıktı değerleri ile gerçek değer arasında oluşan farkın hesaplanmasıyla ortaya çıkmaktadır. Aynı zamanda, tasarlanan sinir ağı modelin başarı oranını da ölçen bir fonksiyondur.

Derin ağların son katmanında maliyet fonksiyonu hesaplanmaktadır. Ağırlık (w) parametresi her katmanda hata oranının düşürülmesi için kullanılır ve maliyet fonksiyonunu azaltmak için her döngüde tekrar hesaplanmalıdır. Denklem 3. 2’de maliyet fonksiyonu hesabı gösterilmektedir.

$$\mathcal{E}(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n)$$

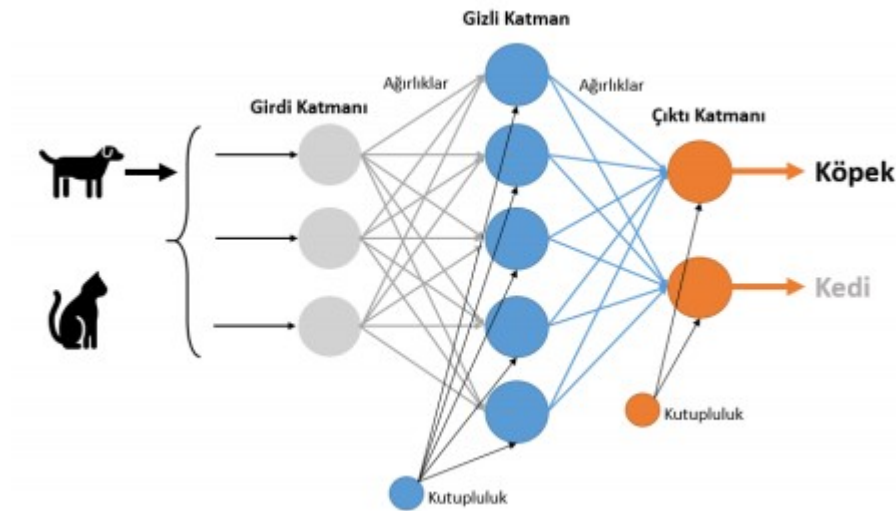
(3. 2)

Ağdaki tüm kayıp, $\mathcal{E}(n)$ ve C çıkış katmanındaki tüm nöronları temsil etmektedir. hata sinyalini ise e_j^2 ifade etmektedir. Ortalama maliyet Denklem 3. 3 ile hesaplanabilir. N değeri eğitim içerisinde bulunan N adet veriyi temsil etmektedir.

$$\begin{aligned}\mathfrak{E}_{av}(N) &= \frac{1}{N} \sum_{n=1}^N \mathfrak{E}(n) \\ &= \frac{1}{2N} \sum_{n=1}^N \sum_{j \in \mathcal{C}} e_j^2(n)\end{aligned}\quad (3.3)$$

3.1.2 İleri Yönde Yayılım (Forward Propagation)

İleri beslemeli sinir ağı, geliştirilen yapay sinir ağının ilk ve en basit türüdür. Katmanlar halinde düzenlenmiş çoklu nöronlar (node, düğüm) içermektedir. Katmanların birbirleriyle bağlantısı bulunmaktadır ve bunlara düğüm denilmektedir. Her düğüm, diğer düğüm için nispi önemi olan bir ağırlığa (w) sahiptir. Şekil 3. 3’de verilen yapay sinir ağı buna örnek olarak verilebilir.



Şekil 3.3. Gizli katmanlara sahip yapay sinir ağı

Yapay sinir ağları verilerin modele verildiği giriş katmanı, özellik haritalamasının yapıldığı gizli katmanları ve sınıflandırma işleminin yapıldığı çıkış katmanı olmak üzere 3 farklı katman yapısına sahiptir. Denklem 3. 4 teki eşitlikten de

anlaşılacağı gibi her katmanın kendine özgü ağırlık ve kutuplama parametre değerleri bulunmaktadır. Bu ağırlıklar girdi değerleriyle çarpılarak bias yani kutuplama değeriyle toplanarak bir sonraki katman için çıktı oluşturur. Oluşturulan çıktıdan önce bir aktivasyon fonksiyonu uygulanır (3.1.5.2). Çıkan değer, bir sonraki katmanın girdisi olacak şekilde model ayarlanmaktadır ve son katmana kadar böyle devam etmektedir. Son katmandaki çıktı değeri ise ağın sonucunu oluşturur.

$$v_j = \sum_l w_{lj}x_l + b_l$$

$$y_j = \sigma(v_j)$$

(3.4)

3.1.3 Geri Yönde Yayılım (Backward Propagation)

3.1.1 de bahsedilen maliyet fonksiyonu hata oranının hesaplanması amacıyla kullanıldığı belirtilmişti. Tüm ağın toplam hata oranı hesaplanıp, bu oranın azaltılması geri yönde yayılım ile gerçekleşmektedir. Toplam hata oranı elde edilen sistemdeki tüm katmanların ağırlık değerlerin yeniden hesaplanması geri yönde yayılım ile sağlanır. Basit bir sinir ağındaki kayıp Denklem 3. 5 ile gösterilebilir:

$$E = (O_{\text{çıkış}(i)} - y_{(i)})^2$$

(3.5)

$O_{\text{çıkış}(i)}$ değeri, bir önceki nöronun değeri ile ağırlık parametresinin çarpımı ve kutuplama değerinin toplamının bir aktivasyon fonksiyonundan geçirilmiş halidir. Bu örnek için aktivasyon fonksiyonu Denklem 3.6'te belirtildiği gibi sigmoid kullanılmıştır:

$$O_{giriş(i)} = w^{(L)} h_{çıkış(i)}^{(L-1)} + b^{(L)}$$

$$O_{çıkış(i)} = \sigma(O_{giriş(i)})$$

(3.6)

$a(L-1)$ değeri, kendinden önceki nörona yani $a(L-2)$ 'ye bağlıdır. Hesaplanmak istenen değer maliyet fonksiyonunun ağırlık değişiminden ne kadar etkilendiğini bulmak için maliyet fonksiyonunun ağırlık değerine göre türevi ($w(L)$) hesaplanmalıdır. Türev kurallarından Denklem 3. 7'deki zincir kuralını uygulanacaktır.

$$\frac{\partial E}{\partial w^{(L)}} = \frac{\partial E}{\partial O_{çıkış(i)}} \frac{\partial O_{çıkış(i)}}{\partial O_{giriş(i)}} \frac{\partial O_{giriş(i)}}{\partial w^{(L)}}$$

(3.7)

Yukarıda belirtilen eşitliklerin Denklem 3.8 ayrı ayrı türevi alınarak eşitlik elde edilir.

$$\frac{\partial E}{\partial O_{çıkış(i)}} = 2(O_{çıkış(i)} - y_{(i)})$$

$$\frac{\partial O_{çıkış(i)}}{\partial O_{giriş(i)}} = \sigma'(O_{giriş(i)})$$

$$\frac{\partial O_{giriş(i)}}{\partial w^{(L)}} = h_{çıkış(i)}^{(L-1)}$$

$$\frac{\partial E}{\partial w^{(L)}} = 2(O_{çıkış(i)} - y_{(i)})\sigma'(O_{giriş(i)})h_{çıkış(i)}^{(L-1)}$$

(3.8)

Bu ifade maliyet fonksiyonunun ağırlık değişimlerinden nasıl etkileneceğini göstermektedir. Hedef, maliyet fonksiyonunu minimumuna ulaşmak olduğu için denklemden değerlerin değişiminin maliyet fonksiyonunu nasıl etkileyeceği

anlaşılabilmektedir. Aynı ifade kutuplama değerleri için de kullanılmaktadır. Olabildiğince basitleştirilmiş olan bu yöntem, yani zincir kuralı bütün ağa uygulanabilir. Daha karmaşık bir yapay sinir ağına uygulandığında bu değişim, Denklem 3.9 ile ifade edilebilir.

$$\frac{\partial E}{\partial w^{(L)}} = \frac{1}{n} \sum_{k=0}^{n-1} \frac{\partial E_k}{\partial w^{(L)}}$$

(3. 9)

Ağırlık değişimleri Denklem 3.10 ile hesaplanabilir:

$$\Delta w_{ij}^k = -\eta \frac{\partial E}{\partial w_{ij}^k}$$

(3.10)

Burada öğrenme katsayısıdır.

3.1.4 Hiper Parametreler

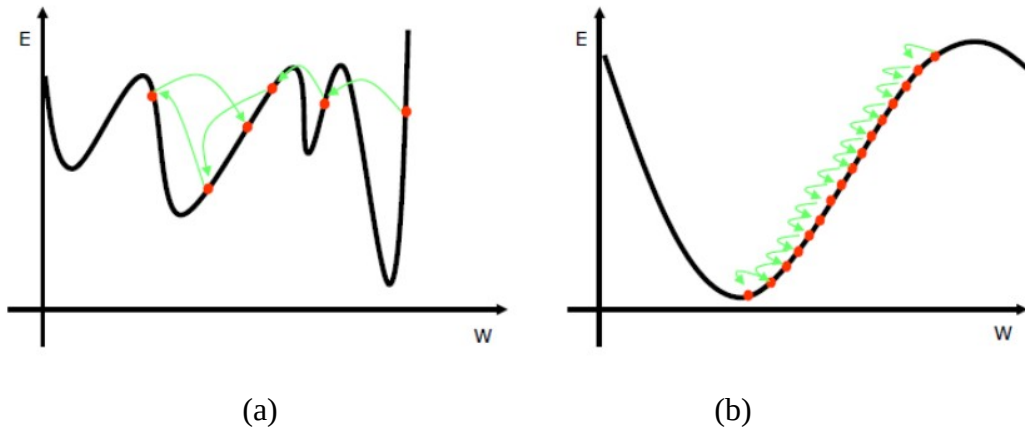
Hiper parametreler, yapay sinir ağlarının eğitilmesi sırasında yüksek başarımlı oranı yakalamasını ve daha az maliyetle gerçekleştirilmesini sağlar. Yapay sinir ağları için kullanılan hiper parametreleri yedi alt başlıkta toplayabiliriz:

- 1- Öğrenme Katsayısı
- 2- Aktivasyon Fonksiyonu
- 3- Eniyileme Türleri
- 4- Ağın Genişliği ve Derinliği
- 5- Epok, Döngü Sayısı ve Paket Boyutu
- 6- Düzenleştirme
- 7- Ağırlık Başlangıç Değerleri

3.1.4.1 Öğrenme Katsayısı

Öğrenme katsayıları, hata oranlarını düşürmek amacıyla kullanılan parametre çeşitleridir ve ağırlıkların değişim miktarını belirler. Şekil 3.10'da görüldüğü gibi optimum bir değerde kullanılmalıdır.

Öğrenme katsayısı gereğinden büyük olursa problem uzayında rastgele gezinmeler olur ve bunun da ağırlıkları katmanlarda rastgele değiştirmekten farkı olmamaktadır. Şekil 3.4.(a)'da verilen grafik bu probleme bir örnektir.



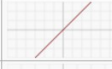

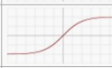






Şekil 3.4.Farklı değerdeki öğrenme katsayılarının sonuç-etki çıktıları

Öğrenme katsayısı çok küçük olursa çözüme ulaşmak daha uzun sürebilmektedir. Şekil 3.4.(b)'de gösterilen örnek üzerinde küçük öğrenme katsayı değeriyle optimum değere doğru ilerlediği ancak yavaş olduğu görülmektedir.

3.1.4.2 Aktivasyon Fonksiyonu

Aktivasyon fonksiyonu, bir katmandan çıktı alındıktan sonra çıkış sinyalinin doğrusal fonksiyon olmasını engellemek amacıyla uygulanmaktadır. Doğrusal fonksiyonlar aynı zamanda tek dereceli polinom olduklarından ağız öğrenimini sınırlandıracakları için aktivasyon fonksiyonu uygulanmaktadır. Şekil 3.5'te aktivasyon fonksiyonları gösterilmektedir.

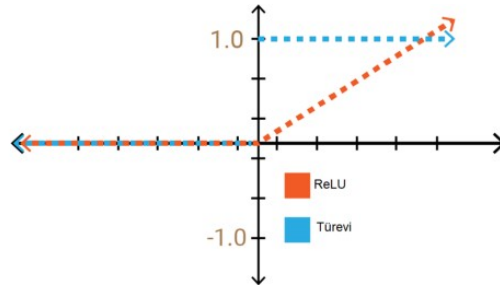
Derin öğrenme modellerinin genellikle birden farklı aktivasyon kullanırken, bu aktivasyon fonksiyonlarının katman kullanımlarına göre seçilmektedirler. Projede kullanılan ResNet18 hazır modelinin bazı katmanlarında ReLu aktivasyon fonksiyonu kullanılmaktadır.

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Şekil 3.5. Aktivasyon fonksiyonları ve türevlenmiş halleri

3.1.4.2.1 Doğrultulmuş Lineer Birim (ReLU) Aktivasyon Fonksiyonu

ReLU, şu anda en çok kullanılan bir aktivasyon fonksiyondur. Neredeyse tüm Evrişimli Sinir Ağları – CNN veya Derin Öğrenme – Deep Learning uygulamalarında kullanılmaktadır.



Şekil 3.6.ReLU fonksiyonu ve türevi

ReLU fonksiyonu Şekil 3. 6'dan da görüldüğü gibi $f(z)$ fonksiyonu, z 'nin negatif değerlerinde çıktı olarak 0; z sıfıra eşit veya pozitif bir değer aldığı anda ise çıktı olarak z değerini vermektedir.

Burada sıfırdan düşük değerde olan her girdi için çıkışta fonksiyon değeri her zaman sıfır olacağı için modelin verilerinin uygun şekilde eğitilme kabiliyetini azaltmaktadır. Negatif değerler için bu fonksiyonun kullanılması verimi düşürecektir. Fakat sağladığı hız ve faydalar sayesinde çokça tercih edilmektedir.

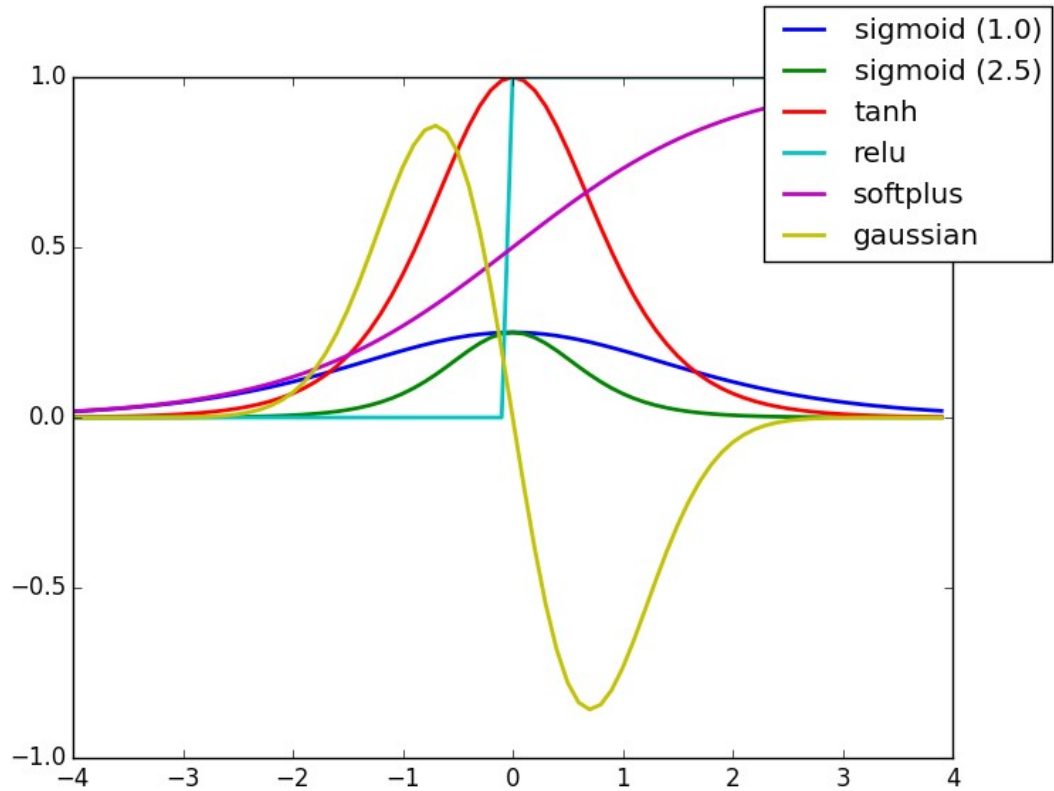
3.1.4.2.2 Aktivasyon Fonksiyonu Seçimi

Her aktivasyon fonksiyonunun kullanım alanları ve amaçları farklı olmakla birlikte, kullanıldığı alanlara göre başarı oranları da değişmektedir. Çok sınıflayıcı fonksiyonlar olarak geniş aralıkta aktive olması için hiperbolik tanjant seçimi uygun olurken; eğitilen modelin biraz daha yavaş öğrenilmesi istenmesi durumunda sigma fonksiyonu kullanılabilir.

Kullanılan modelde ağ derinse ve işlem yükü önemli bir problem halinde ise ReLU aktivasyon fonksiyonunun tercih edilmesi daha doğru bir karar olacaktır.

ReLU fonksiyonu LeakyReLU'ya göre daha hızlı işlem yapmaktadır ancak ReLU'daki negatif değerlerin sifıra eşitlenmesi sorununa çözüm olarak da LeakyReLU aktivasyon fonksiyonunu kullanmak doğru bir karar olacaktır.

Kullanacağımız veri setindeki işlem yükünden dolayı ReLU aktivasyon fonksiyonu en optimize sonucu vermektedir.

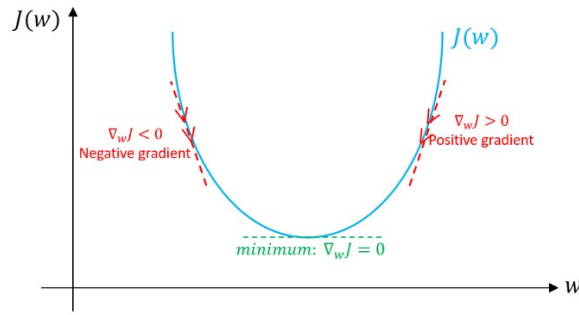


Şekil 3.7. Aktivasyon fonksiyonlarının türevleri

Backward Propagation (Geriye Yayılım) Modeli'nin uygulanması sırasında, kayıp fonksiyonunun ve diğer parametrelerin uygun kullanılabilmesi için aktivasyon fonksiyonları türevlenebilir olmalıdır. Bu ağın öğrenmesini etkiler. Şekil 3.7'de görüldüğü üzere ReLU aktivasyon fonksiyonlarının türevlerinin de eğitim sürecinde benzer bir grafiği bulunmaktadır ve bu da öğrenimdeki başarımların artırılmasında rol oynar.

3.1.4.3 Eniyileme Türleri

Ağın ürettiği çıkış değeri ile gerçek değer arasındaki farkı azaltmak için kullanılan yöntemdir. Yani maliyet fonksiyonunun değerini azaltmak istenir. Genellikle eğim düşümü (Gradient Descent) kullanılarak bu optimizasyon işlemi gerçekleştirilir.

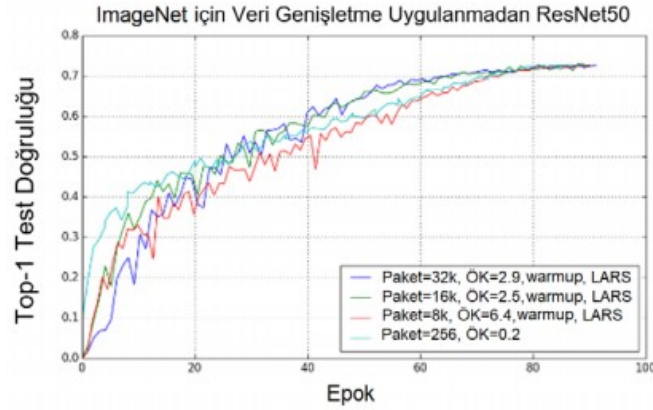


Şekil 3.8. Eğim düşümünün uygulanması

Şekil 3.8'deki eğim düşümü maliyet fonksiyonunun azaltılmasıyla (eğimin azaltılmasıyla) gerçekleşmektedir.

3.1.4.4 Epok, Döngü Sayısı ve Paket Boyutu

Epok, veri setinden alınan paketlerin ileri ve geri yayılım ile ağın tamamından geçmesidir. Paket boyutu, ileri ve geri yöndeki yayılım için veri setinden alınan/kullanılan veri miktarıdır. Paket boyunun yüksek seçilmesi, iterasyonun için kullanılacak eğitim (training) örneğinin de büyümesi anlamına gelmektedir. Epok sayısının artırılması her zaman doğruluğu/başarıyı arttıracak anlamına gelmez. Gereğinden fazla epok sayısı modelde aşırı öğrenme sorununu tetikler.



Şekil 3.9.Doğruluk ve epok sayısının arasındaki ilişki

Şekil 3.9’da görüldüğü gibi epok sayısının artışı belirli bir noktadan sonra doğrulukta bir etki etmediği gözlemlenmektedir. Ağın doğruluğunu etkileyen bir diğer faktör de paket boyutudur. Paket boyutunun çok fazla artması doğrulama hatasının da çok hızlı artmasına neden olmuştur.

3.2 Evrişimli Sinir Ağları

Bu bölümde neden derin öğrenme ve evrişimli sinir ağlarının sistemimizde kullanıldığından bahsedilmiş ve kullanılan teknikler açıklanmıştır.

3.2.1 Derin Öğrenme

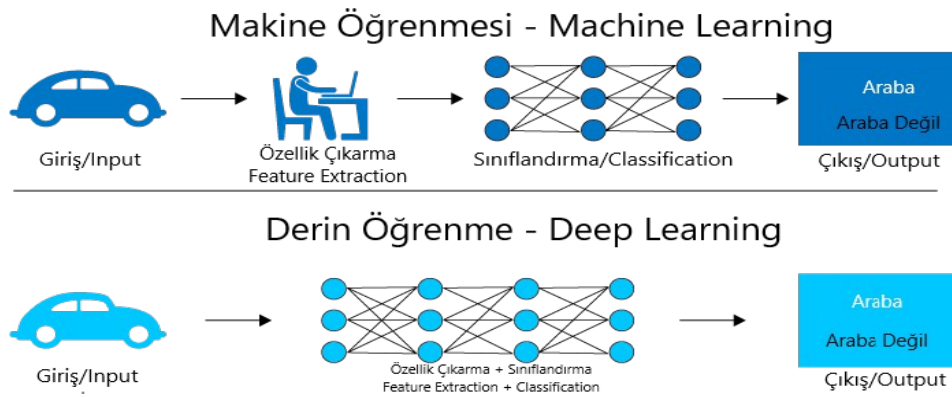
Son yıllarda endüstri ve akademik alanlardaki veri bilimciler görüntü sınıflandırma, video analizi, konuşma tanıma ve doğal dil öğrenme süreci dahil olmak üzere çeşitli uygulamalarda gelişmeler elde etmek için makine ile öğrenmede GPU’ları (Grafik İşlemci Ünitesi) kullanmaktadırlar. Üniversiteler ve özel kuruluşlar büyük miktarlarda eğitilmiş veri setlerini açık kaynak haline getirerek ileri seviye teknolojilerin geliştirilmesinde katkı sağlamaktadırlar. Çok seviyeli “derin” sinir ağlarının kullanılması olarak tanımlanan Derin Öğrenme, önemli derecede yatırım ve

araştırmanın yapıldığı bir alandır. Geline bu noktada, milyarlarca parametreyi hesaplamak için veriden farklı bilgiler çıkarmak gerekmektedir. Görüntü Sınıflandırma (Image Classification), Nesne Bulma (Object Detection), Nesne Takip Etme (Object Tracking), Doğal Dil İşleme (Natural Language Processing), Stil Transferi (Style Transferring) gibi problemlerin çözümü yapay sinir ağı tabanlıdır.

GPU ile paralel hesaplama yöntemi bu elde edilen büyük veri setlerinin kullanımını kolaylaştırmakta ve derin nöral ağları çok daha kısa sürelerde, daha az veri altyapısı kullanarak eğitilmektedir. CPU'lara (Merkezi İşlem Birimi) kıyasla GPU'ların 10 ile 100 kat uygulama performansı sundukları gözlemlenmektedir.

Dolayısıyla GPU'lar ile önceden kaydedilen multimedya içerikleri çok daha hızlı bir şekilde yazıya geçirebilmektedir.

Makine Öğrenmesinin bir alt dalı olan Derin Öğrenme, klasik makine öğrenme metodlarına kıyasla çok daha fazla veriyi işleyebilmektedir. Ayrıca Makine Öğrenmesinde bazı işlemlerin(özellik çıkarma gibi) manuel olarak yapılmasının karşısında; Derin Öğrenmede derin sinir ağları -parametrelerin belirlenmesiyle birlikte- eğitilerek modeller elde edilmektedir.



Şekil 3.10.Makine öğrenmesi ve derin öğrenme modelleri

Şekil 3.10’da derin öğrenme ve makine öğrenmesi karşılaştırılmıştır. Buradan anlaşılacağı üzere bilgisayarların kendi kendilerine öğrenmeleri derin öğrenme ile mümkündür. Belirli hata katsayılarının elde edilip, yine aynı hata katsayıların azaltılması derin öğrenme modeli ile mümkündür.

3.2.2 Evrişimli Sinir Ağları (Convolutional Neural Networks-CNN)

Evrişimli Sinir Ağları, evrişimsel ağlar (LeCun, 1989) olarak da bilinir, ızgara benzeri bir topolojiye (grid-liketopology) sahip olan ve verilerin işlenmesi için uzmanlaşmış bir tür sinir ağıdır. Girdi olarak aldığı görüntüyü daha önce eğitildiği veriler üzerinden özellik çıkarma ve sınıflandırma yaparak tanımlar. Özellikle ayırım/sınıflandırma yapmak amacıyla kullanılmaktadır.

Derin öğrenmenin alt dalı olan Evrişimli Sinir Ağları, girdilerden öznitelik çıkarımı ile sınıflandırma, tahmin gibi farklı işlemler için kullanılır. Genel bir Evrişimli Sinir Ağlarında “Girdi katmanı”, öznitelik çıkarımı yapılan birden fazla katmana sahip olabilen “Evrişimli katman” ve sınıflandırma işleminin gerçekleştiği “Tam bağımlı katman”dan oluşur. Şekil 3.10’da derin öğrenme kısmında örnek bir ESA görülmektedir.

Yıl	Algoritma	Başarım (Top-5 Doğruluk)
2012	AlexNet	%84,7
2013	ZFNet	%85,2
2014	VGGNet	%92,7
2014	GoogLeNet	%93,33
2015	ResNet	%96,43
2016	Trimp	%97.01
2017	SENet	%97,75

Şekil 3.11 2012-2017 yılları arasında dereceye giren ağların hata-doğruluk oranları

Şekil 3.11 incelendiğinde 2012 yılından itibaren evrişimli sinir ağlarının başarımlarında bir artış söz konusudur. Son zamanlarda resim tanıma doğruluk oranı,

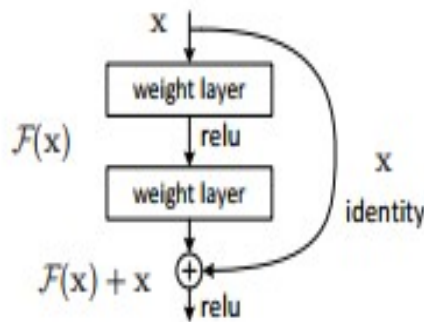
sınırlandırılmış olan insan algısına yaklaşmıştır. Evrişimli sinir ağları ile eğitilen modeller artık günlük hayatımızda yer almaya başlamıştır.

3.2.2.1 Artık Sinir Ağı (Residual Neural Network – ResNet)

Şekil 3.11’de gösterilmekte olan derin öğrenme modelleri içerisinde de bulunan ResNet, 2015 yılında He Kaiming, Sun Jian ve Microsoft Research Asia’den diğerleri tarafından önerilen bir ağ yapısıdır ve ILSVRC-2015 sınıflandırma görevinde birinci olmuştur. Aynı zamanda ImageNet algılama, ImageNet yerelleştirme, COCO algılama ve COCO segmentasyon görevlerinde birinci olmuştur.

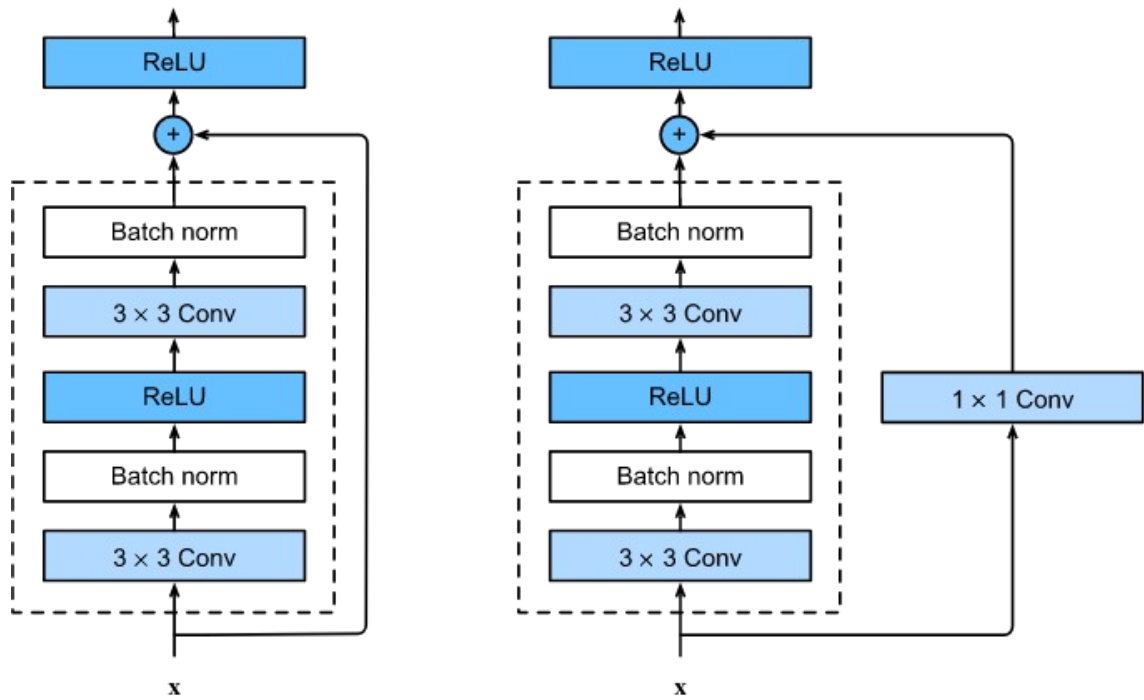
Artık/Artan sinir ağı, serebral korteksteki piramidal hücrelerden bilinen yapılar üzerine inşa edilen bir tür yapay sinir ağıdır.

Derin ağ modellerinde kullanılan mantıktan daha farklı bir mantığa sahip olan artık sinir ağı, artık değerlerin (residual value) sonraki katmanları besleyen artık bloklara (residual blok) eklenmesiyle oluşmaktadır. Bu yapısı ile diğer modellerden farklılık sağlamaktadır. Doğrusal ve ReLu arasındaki iki katmanda bir eklenen bu artık değer, Şekil 3.12’deki görselde gösterildiği gibidir. $a[l]$ artık değeri iki katman sonraki $a[l+2]$ hesabına eklenmektedir ve sondaki katmana artık değer eklenerek artık blok değeri $F(x)$ yerine $F(x) + x$ olarak hesaplanır.



Şekil 3.12. ResNet yapısı, $\ell - 1$ katmanı, $\ell - 2$ 'den etkinleştirmeye atlanır.

Derin ağ modellerinde, katmanların parametre olarak kullandığı ağırlıkların derinlik katmanlarına doğru eğimin sıfıra yaklaşması problemi kaybolan eğim/gradyan (vanishing gradient) olarak adlandırılmaktadır. Bunun kaybın olmasının nedeni, ağ derinlere ilerledikçe kayıp fonksiyonunun hesaplandığı eğimlerin sıfıra doğru inmesidir. Ağırlık değerleri bu eğim probleminden sonra güncellenememesi ve bu doğrultuda öğrenme yapamaması söz konusu olmaktadır. Artık sinir ağı katmanlar arasında blok değerler aktararak bu problemin önüne geçmektedir.



Şekil 3.13. 1x1 Evrişim içeren ve içermeyen ResNet bloğu

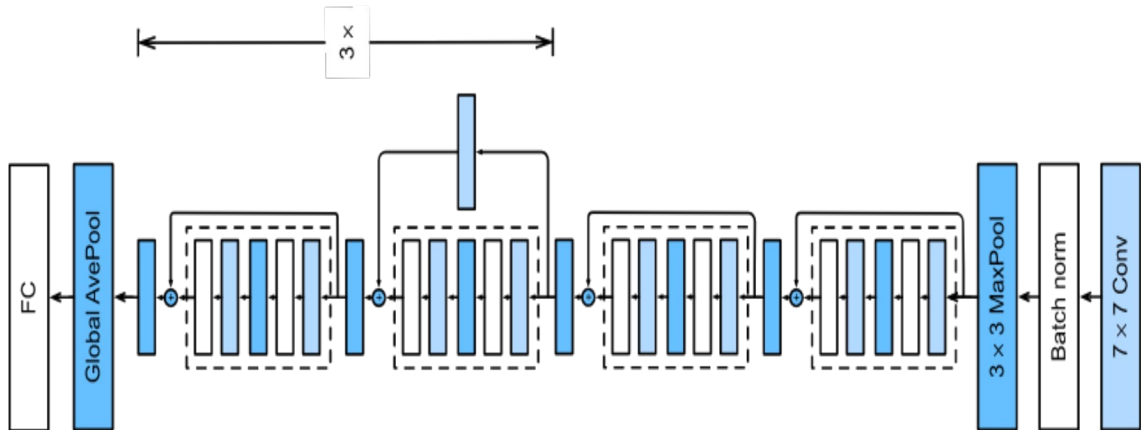
Şekil 3.13'teki Girdiyi x ile belirtildiğini düşürsek, öğrenerek elde etmek istenilen alta yatan eşlemenin üstteki etkinleştirme işlevine girdi olarak kullanılacak $f(x)$ olduğunu varsayılmaktadır. Şekil 3.13'ün solunda, kesik çizgili kutudaki kısım doğrudan $f(x)$ eşlemesini öğrenmelidir. Sağda ise, kesik çizgili kutudaki bölümün, artık bloğun adını belirleyen *artık eşleme* $f(x)-x$ 'i öğrenmesi gerekmektedir. Birim eşlemesi $f(x)=x$ istenen alta yatan eşleme ise, artık eşlemeyi öğrenmek daha kolaydır: Sadece

kesik çizgili kutudaki üst ağırlık katmanının ağırlıklarını ve ek girdilerini (örn. tam bağlı katman ve evrişimli katman) sıfıra itmemiz (vanishing gradient - sıfıra yakınsama) gerekir.

Artık sinir ağının artık bloğu, Şekil 3.12’de sağ tarafta bulunan şekilde gösterilmektedir. Burada x katman girdisini toplama işlemine taşıyan düz çizgiye artık bağlantı (veya kısayol bağlantısı) denir. Artık bloklarla girdiler, katmanlar arasında kalan bağlantılar üzerinden daha hızlı yayılabilir[11].

3.2.2.1.1 ResNet18 ve ImageNet

ResNet18, diğer artık sinir ağı modellerine kıyasla daha basit olan ve bu basitliğinden dolayı daha hızlı çalışan bir artık yapay sinir ağı modelidir. Tez çalışması sürecinde bu model kullanılmaktadır. ImageNet, hiyerarşinin her bir düğümünün yüzlerce ve binlerce resimle gösterildiği WordNet hiyerarşisine (şu anda sadece isimler) göre organize edilmiş bir görüntü veri setidir. Artık sinir ağı mimarisi bu veri veri yapısına göre oluşturulmuş olup, toplamda çıkışında 1000 adet sınıf bulundurmaktadır.



Şekil 3.14. ResNet-18 Mimarisi

3.2.3 Evrişimli Sinir Ağları Mimarisi

CNN mimarisi genellikle girdi katmanı, evrişim katmanı, havuzlama (pooling) katmanı, tam bağlaşımlı katman (fully-connected layer) ve çıktı katmanlarını bulundurmaktadır.

3.2.2 bölümünde eğitilen veri setlerinden bahsedilmişti. Aşağıda veri setlerinin nasıl eğitildiğine dair açıklama yapılmıştır.

3.2.3.1 Evrişim Katmanı

Evrişim özelleştirilmiş olan bir doğrusal işlemdir. Matris çarpımı gibi kompleks ve uzun süre alacak işlemler yerine evrişim işlemini gerçekleştiren bu ağlar, daha kısa sürede sonuç vermektedir. Ayırık zamanlı evrişim işlemi Denklem 3.11’de verilmiştir:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

(3.11)

3.11 numaralı denklemde girdi x , filtre (kernel) w , zaman t ve sonuç s olarak belirtilmiştir. Girdi olarak resim gibi iki boyutlu bir girdi kullanıldığında evrişim işlemi, 3.12 numaralı denklemde verilmiştir.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i,j)K(i-m,j-n)$$

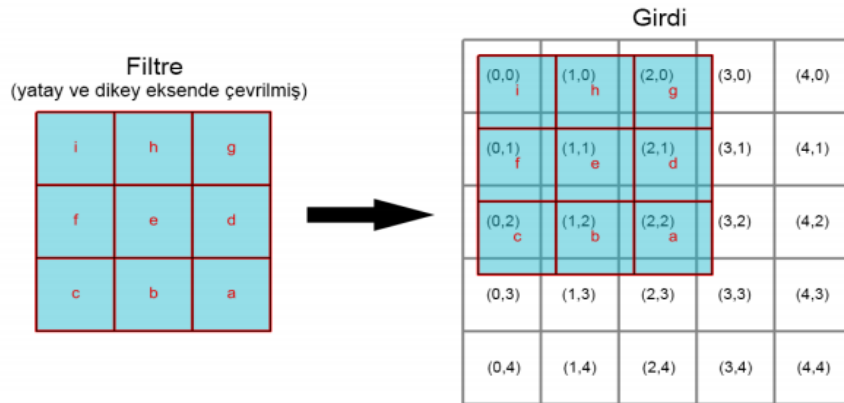
(3.12)

3.12 numaralı eşitlikte i ve j terimleri, evrişim sonucu elde edilen matrisin konumlarını ifade ederken; m ve n terimleri de filtrenin konumunu ifade etmektedir. Filtrenin merkezi genellikle orijinde olacak şekilde konumlandırılır. Şekil 3.15'teki örnek filtrenin merkez elemanı orijinde konumlandırılmıştır. Bunun amacı işlem kolaylığı sağlamaktır. Filtre elemanı $(0,0)$ noktasında bulunmaktadır.

m \ n	-1	0	1
-1	a	b	c
0	d	e	f
1	g	h	i

Şekil 3.15. 3x3 Filtre

Şekil 3.16'da bir girdi verilmiştir. Bu girdiye filtre ile evrişim işlemi uygulanırsa, çıktının $(1,1)$ konumundaki değeri 3.13'deki eşitlik ile hesaplayabilmektedir.



Şekil 3.16 Evrişim işlemi örneği

$$\begin{aligned}
 (I * K)(1,1) = & I(0,0)K(1-0,1-0) + I(1,0)K(1-1,1-0) \\
 & + I(2,0)K(1-2,1-0) + I(0,1)K(1-0,1-1) \\
 & + I(1,1)K(1-1,1-1) + I(1,2)K(1-1,1-2) \\
 & + I(0,2)K(1-0,1-2) + I(1,2)K(1-1,1-2) \\
 & + I(2,2)K(1-2,1-2)
 \end{aligned}$$

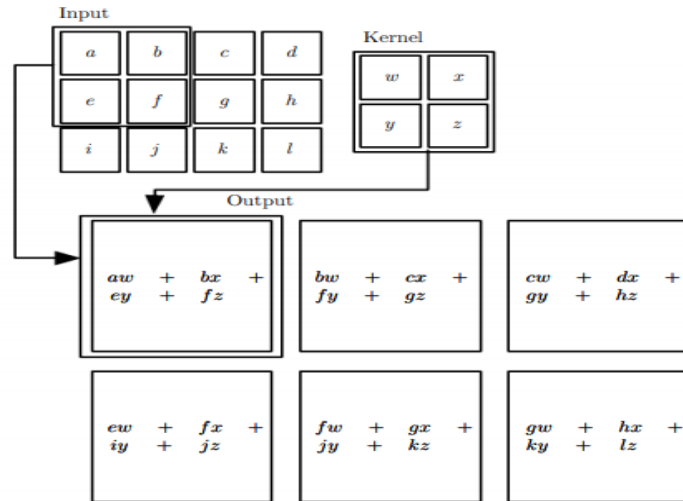
(3.13)

Şekil 3.16’da uygulanan evrişim örneğindeki filtre çevrilmiş halde girdiye uygulanmaktadır. Çevirme işlemleri yatay veya dikey olarak gerçekleştirilebilmektedir. Bunun yerine, çoğu sinir ağı kütüphaneleri, evrişim ile aynı olan ancak çekirdeği/filtreyi (kernel) çevirmeden “Çapraz Korelasyon(Cross-Correlation)” adı verilen ilgili bir fonksiyon uygular(3.14):

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

(3.14)

Şekil 3.17’de çapraz korelasyon ile evrişim işleminin uygulanması üzerine bir örnek verilmiştir.



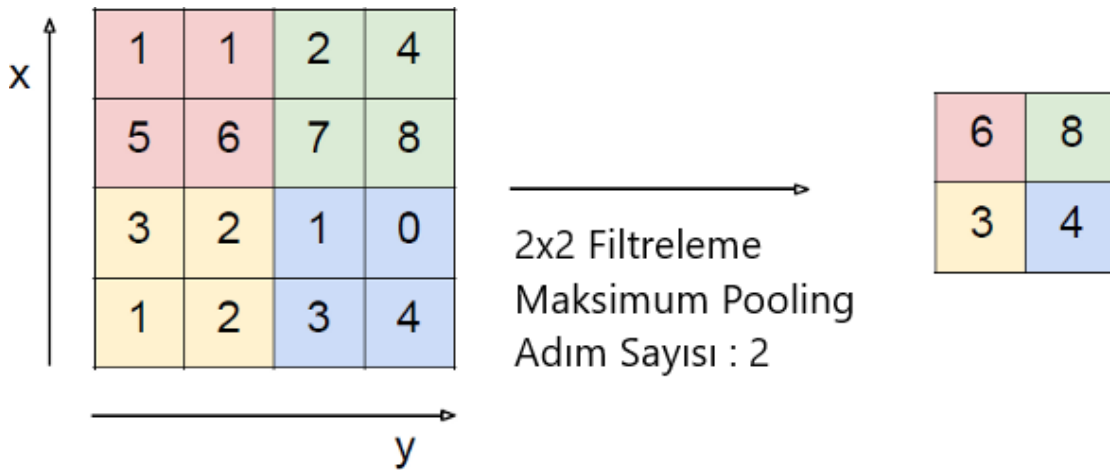
Şekil 3.17. Çaprazkorelasyon fonksiyonu ile evrişim işlemi

Şekil 3.17’de görüldüğü gibi 2x2’lik bir örnek filtre girdi üzerinden kayarak ilerlemektedir. Filtre her yeni konumuna geldiğinde evrişim işlemi devam eder ve çıktılar üretilir. Üretilen çıktılar yeni bir matrise yazılır. Buradaki örnekte 4x4’lük bir girdi matrisi ve 2x2’lik filtre matrisi verilmiştir. Filtre girdi üzerinde adım sayısı 1 adım olacak şekilde kayarak evrişim işlemini yapmıştır ve yeni oluşan matrisin boyutu da 3x3’tür. Filtre gezdirme işlemi farklı boyutlardaki matrislerle de yapılabilir.

3.2.3.2 Havuzlama/Ortaklaşma Katmanı

Giriş matrisinin kanal sayısını sabit tutarak yükseklik ve genişlik bilgisini azaltır. Hesaplama karmaşıklığını azaltmak için kullanılan bir adımdır. Ancak Hinton’ın kapsül teorisine göre verideki bazı bilgilerin de kaybolmasına sebep olduğu için başarımdan ödün vermektedir. Bu katmanda genellikle maksimum ortaklaşma yöntemi kullanılır. Ağın bu katmanında öğrenilen parametre yoktur.

Genellikle maksimum ortaklaşma yöntemi kullanılmaktadır. Özellikle konum bilgisinin çok önemli olmadığı yerlerde güzel sonuçlar vermektedir.



Şekil 3.18. Ortaklaşma işlemi gösterimi

Şekil 3.18’de 2x2 maksimum-ortaklaşma filtresi, adım sayısı 2 olarak kullanılmıştır. İlgili 4 elemanın olduğu alandaki en büyük değer çıkışa aktarılır. Çıkışta 4’te 1 boyutlu bir veri elde edilmiş olur.

3.2.3.3 Tam Bağlaşımlı Katman

Yapay sinir ağları gibi çalışmakta olan bu katmanda, ortaklaşma/havuzlama ve evrişim işlemleri sonucunda üretilen değerler girdi olarak alınıp işleme sokulmaktadır. Çıkış katmanında sınıf sayısı kadar sonuç üretilmektedir.

3.2.4 Evrişimli Sinir Ağlarında Dolgu Ekleme ve Adım Sayısı

Evrişim işlemi uygulanan görüntü piksellerinin eğitimi sırasında özellik çıkarma önemli bir aşamadır. Sistem katmanları sonunda sınıflandırmanın sağlanması için kullanılan bu yöntem ile birlikte ağ, girdilerin farklarını bulabilmektedir. Bu işlemlerdeki matrislerin boyut uyumunun sağlanması ve başarı oranının yükseltilmesi için girdi matrislerine dolgu ekleme (padding) yapılmaktadır. Ayrıca, girdi matrisleri üzerinde evrişim işlemi de belirli bir adım sayısı (filtre matrisinin girdi matrisi üzerinde kaydırılması) ile yapılmaktadır.

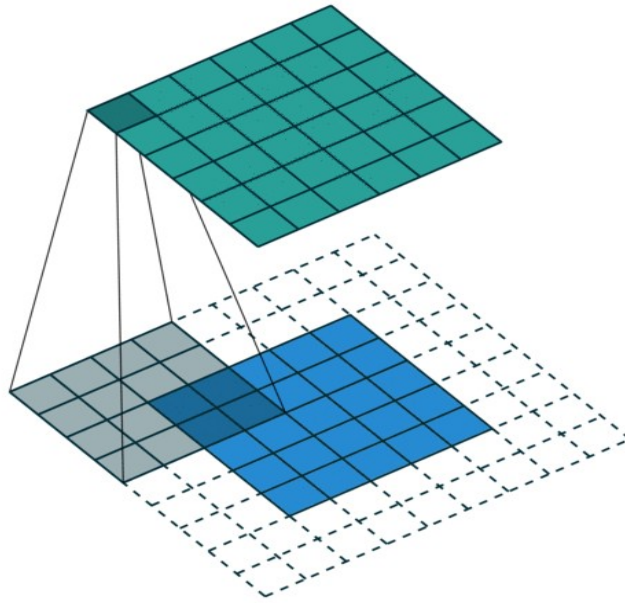
3.2.4.1 Dolgu/Piksel Ekleme

Evrişim işleminden sonra giriş işareti ile çıkış işareti arasında boyut farkını yönetmek elimizde olan bir hesaplama değildir. Bu işlem giriş matrisine eklenecek ekstra pikseller ile sağlanır. Bu yapılan işlem “dolgu/piksel ekleme” olarak adlandırılmaktadır. Giriş matrisi $n \times n$, filtre (ağırlık) matrisi $f \times f$ olduğu durumlarda çıkış matrisinin giriş matrisi ile aynı boyutlu olması isteniyorsa eşitlik 3.15’teki formül uygulanır.

$$(n+2p-f+1)*(n+2p-f+1) \quad (3.15)$$

3.15 numaralı eşitlikte belirtilen ‘p’, giriş matrisine eklenen piksel boyutudur yani dolgu (padding) değeridir ve bunu belirlemek için 3.16 numaralı denklemden faydalanılır.

$$p=(f-1)/2 \quad (3.16)$$



Şekil 3.19. Girdi matrisine dolgu/piksel ekleme örneği

Şekil 3.19’da gösterilen piksel ekleme işleminde giriş matrisine eklenilen [2,2,2,2] pikselleri sonucunda çıkış matrisi (5x5), giriş matrisine (5x5) eşit olmuştur.

3.2.4.2 Adım Sayısı

Adım Sayısı (Stride), evrişim işlemi için ağırlık matrisi olan filtreyi görüntü üzerinde birer piksellik adımlarla ya da daha büyük adımlarla kaydıracağının bilgisini verir. Bu da doğrudan çıkış boyutunu etkileyen diğer bir parametredir.

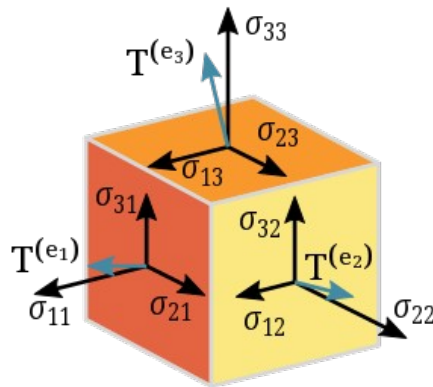
$$\left[\frac{n+2p-f}{s} + 1 \right] \times \left[\frac{n+2p-f}{s} + 1 \right] \quad (3.17)$$

Denklem 3.17’de girdi (nxn), filtre (fxf), dolgu ekleme (p), adım sayısı (s) ile gösterilmektedir ve çıktı matrisinin boyutunu belirtmektedir.

3.2.5 PyTorch ve Tensörler

PyTorch, Python tabanlı ve grafik işlem birimlerinin (GPU) gücünü kullanan bir bilimsel bilgi işlem paketidir. Facebook firması tarafından geliştirilmiş, derin öğrenme uygulamalarında kullanılan, maksimum esneklik ve hız sağlamak için geliştirilmiş araştırma platformu, yazılım kütüphanesidir.

PyTorch’un üst düzey özelliklerden ikisi; güçlü GPU hızlandırma desteğiyle tensör hesaplamaları ve teyp tabanlı bir otograd sistemlerinde derin sinir ağları oluşturmaktır.



Şekil 3.20. Tensör yapısı

Tensör, çok boyutlu bir verinin simgelenebildiği geometrik bir nesnedir[12]. Öklid mesafesinin karesini en aza indirerek x 'ten y 'yi tahmin etmek üzere eğitilmiş, tek bir gizli katmana sahip ve önyargı içermeyen tamamen bağlı bir ReLU ağı. Bu uygulama, ileri geçiş, kayıp ve geri geçişi manuel olarak hesaplamak için PyTorch tensörlerini kullanır.

Bir PyTorch Tensörü temelde bir numpy dizisiyle aynıdır: derin öğrenme veya hesaplamalı grafikler veya gradyanlar hakkında hiçbir şey bilmez ve yalnızca rastgele sayısal hesaplama için kullanılacak genel bir n -boyutlu dizidir. Numpy kütüphanesinden oluşturulmuş bir dizi ile bir PyTorch Tensor arasındaki en büyük fark, bir PyTorch Tensor'un CPU veya GPU üzerinde çalışabilmesidir. İşlemleri GPU'da çalıştırmak için Tensor'u bir CUDA veri türüne aktarmak yeterli olacaktır.

BÖLÜM 4. PROJE MODELİNİN OLUŞTURULMASI

Bu bölümde proje kapsamında elde edilen veri setinin kullanılması amacıyla, ResNet modelinin baz alındığı ve PyTorch kütüphanesinin kullanılarak oluşturulan modelin ve yazılan kodun aşamalarından bahsedilmektedir.



Şekil 4.1. Modele girdi olarak verilen görüntülerin model tarafından tahminlenmesi

Proje Kapsamında aşağıdaki aşamalar gerçekleştirilmiştir:

Aşama 1: Kütüphanelerin Girilmesi ve Veri Setinin Oluşturulması

Aşama 2: Görüntü Dönüşümleri

Aşama 3: Veri Yükleyici (DataLoader) Hazırlama

Aşama 4: Veri Görselleştirme

Aşama 5: ResNet ile Modelin Oluşturulması

Aşama 6: Modelin Eğitilmesi

Aşama 7: Sonuçlar ve Öneriler

4.1 Kütüphanelerin Girilmesi ve Veri Setinin Oluşturulması

Modelin oluşturulmasından önce görüntülerin bulunduğu klasörlerden çekilerek modele sunulması ve işlemlerin yapılması için Python programlama dilinin kullanılması

gereken kütüphaneler bulunmaktadır. Bunlar; os, shutil, random, torch, torchvision, numpy, PUL ve matplotlib kütüphaneleridir.

Veri setlerinde bulunan görüntülerin klasörlerden bulunmasını, liste halinde çekilmesini ve matrissel dizilerde tutularak matematiksel işlemlerle görüntüler üzerinden özellik haritası çıkarılarak modele sınıflandırılmanın yaptırılması genel amaçtır.

Train ve Test başlıkları altında konumlandırılan veri setlerinin öncelikle elde edildiği konuma bağlı olarak sınıflarının (layer) da tanımlanması ve bölüm 4.3'teki veri yükleyicide dönüştürülmek üzere hazırlanması gerekmektedir. Hazır veri seti normal, viral ve covid olacak şekilde sınıflandırıldığı için katman adları bu şekilde olacakken, bölüm 4.2'deki görüntülerin dönüştürülme formatları ile birlikte modele yüklenecektir. Burada veri setinden çekilen görüntüler, direkt olarak pytorch ile oluşturulan modele aktarılamamaktadır. ImageNet veri setine göre daha önce eğitilmiş olan (pre-trained) ResNet18 modeli, 3 kanallı görüntü girdisi beklediği için veri setindeki görüntülerin RGB formata dönüştürülmesi sağlanmıştır.

4.2 Görüntü Dönüşümleri

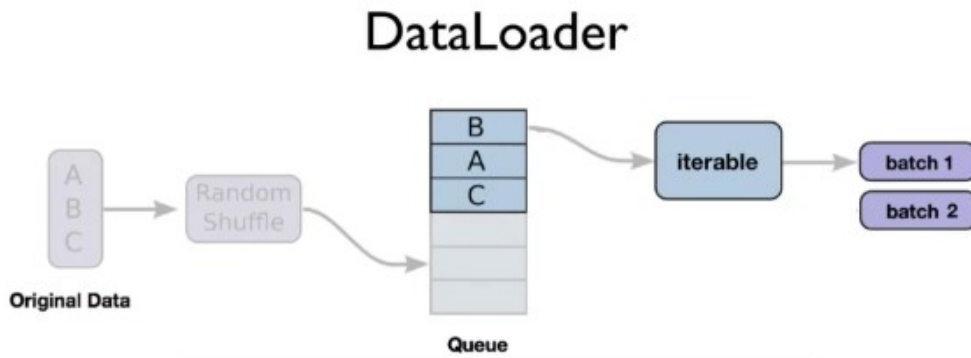
Eğitim (Train) ve test veri setlerindeki görüntülerin dönüşümlerinin yapıldığı bölümdür. Modelin kabul edeceği şekilde görüntülerin boyutları (224,224) formatında değiştirilmektedir. Eğitim bölümündeki görüntülerin özellik haritasında (feature map) daha baskın bir farklılıklarının görüntülenmesi için yatay bazlı görüntülerin çoğaltılması gerçekleştirilmiştir. Şekil 4.2'de pytorch kütüphanesinin hazır bir metodu olan rastgele yatay değiştirme (random horizontal flip) ile veri setindeki eğitilecek görüntülerin çoğaltılması sağlanmaktadır. Bu işlem yalnızca eğitilecek olan görüntülere yapılmaktadır.



Şekil 4.2. Verisetinin çoğaltılması için görüntülerin yatay formatta değiştirilmesi

Bölüm 4.3'teki verilerin modele yüklenmesi öncesinde ResNet18 modeline uygun olarak görüntünün oluşturulması için tensörler haline çevrilen görüntülerin yine ResNet18 modeline uygun olacak şekilde normalizasyon işlemlerinden geçmesi gerekmektedir. Bu işlem sonrasında görüntünün dönüştürülmesi tamamlanmış bulunmaktadır.

4.3 Veri Yükleyici (DataLoader) Hazırlama



Şekil 4.3. Veri yükleyici genel yapısı

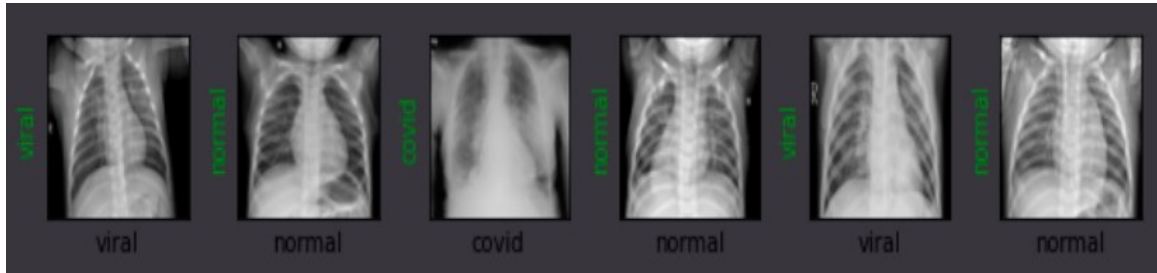
Bölüm 4.1 ve 4.2'deki işlemler sonrasında veri seti görüntülerinin hazırlanmasından sonra modele verilmesi gerekmektedir. PyTorch, bir veri kümesindeki verileri bir modül aracılığıyla ('ingest') alma mekanizmasına sahiptir ve bu veri yükleyici (Data Loader) olarak bilinir. Bu veri yükleyici, derin öğrenme modeline elde edilen verileri basar. Bu

bölümde ayrıca verilerin paketler halinde (batch size) basılması için gerekli olan tanımlama yapılmaktadır. Şekil 4.3'te veri yükleyicinin genel yapısı verilmiş olup, almış olduğu veriyi iterasyonlarla partilere/paketlere bastığı görülmektedir.

4.4 Veri Görselleştirme

Modele basılmadan önce eğitim ve test veri setlerinin veri yükleyici üzerine yüklenmesinin ardından, model eğitimi öncesi ve sonrası olmak üzere rastgele gelen görüntülerin görüntülenmesi gerçekleştirilir. Bu aşamada asıl amaç, modele basılmadan önce veriler üzerinde herhangi bir problem olup olmadığının teyit edilmesi yapılmaktadır. Elde edilen çalışma üzerinde varsayılan parti boyutu (batch size) 6 olduğundan, veri setinden de 6'şar adet görüntü rastgele çekilerek, kontrol edilmektedir.

Veri setinin modele verilmesinin ardından da verilerin görselleştirilmesi işlemi, modelin çıktılarının karşılaştırılması amacıyla kullanılmaktadır.



Şekil 4.4. : Veri setlerinden rastgele elde edilen verilerin görselleştirilmesi

Üç farklı sınıf üzerinden eğitilen modelin daha sonrasında test veri seti üzerinden sonuçlarının görüntülenmesi için kolaylık sağlamaktadır. Modelden elde edilen rastgele görseller bu metod üzerinden geçirilirken modelin test veri setleri hakkında tahminleri ve veri setinin asıl sınıf şablon değeri de parametre olarak verilirken, karşılaştırma sonrasında yeşil-doğru tahmin; kırmızı-yanlış tahmin olmak üzere görselleştirme yapılmaktadır. Modele girdi olarak sunulacak verilerin görselleştirilme işlemi Şekil 4.4'deki gibi yapılmaktadır.

4.5 ResNet18 ile Modelin Oluşturulması

Bölüm 3.2.2.1’de yer alan artık/artan sinir ağı modeli, PyTorch kütüphanesinde bulunmaktadır. Bu kütüphane modelin tüm katmanlarını içerisinde barındırırken, kullanıcılara da her bir katmanındaki parametreleri varsayılan şekilde kullanma ve değiştirme imkanı sağlamaktadır.

Torchvision adlı kütüphanesi ile yöntemi kullanılarak model katmanları elde edilmiştir. ImageNet adlı veri setini varsayılan olarak kullanan ResNet18 ağ modeli, kullandığı bu veri setinde toplamda 1000 adet farklı sınıf için tasarlanmıştır. Elde edilen radyoloji veri seti toplamda 3 adet farklı sınıflardan oluştuğu için modelin doğrudan kullanılabilir değildir. Model, bulundurduğu sınıf sayısının varsayılanda 1000 adet olması, model sonundaki tam bağlantılı katman (fully connected layer) çıktısının da 1000 adet olduğunu göstermektedir. 3 adet olan modelimize uygun olarak tam bağlantılı katman sayısını üçe düşürülmüş bulunmaktadır. Böylelikle hazır olan modelin, kendi modelimize dönüştürülmesi sağlanmışken, herhangi bir şekilde parametre sayısının değiştirilmesine gerek duyulmamıştır. Optimize parametresi olarak adam_optimizer kullanılmıştır. ResNet18 model mimarisi ile ayrıntılı bilgiler Bölüm 3.2.2.1.1’de ayrıntılı olarak açıklanmıştır.

Modelin eğitiminden önce, resnet18’in görüntü sınıflandırma yöntemi kullanarak çıktı öncesinde karşılaştırma verileri elde edilmiş ve model tahminlemesi yapılmıştır. Şekil 4.5’te görüldüğü üzere, modele girdi olarak sunulan verilerin tahminlenmesi tamamen rastgele ve yetersiz bir oranda (%40 altında) doğruluktadır. Bunun nedeni modelin henüz veri setini öğrenmemiş olmasıdır. Model öğrendikçe daha doğru tahminlemeler yapması beklenmektedir.



Şekil 4.5. Model eğitiminden önce ResNet ağ modelinin tahminleme çıktıları

4.6 Modelin Eğitilmesi

Modelin eğitiminden önce veri setini, yükleyici üzerinden girdi olarak modele verdik. Şekil 4.6’da özet görüntüsü verilen model ile eğitime başladık. Eğitimde epok sayısını 10 olarak ayarladık. Eğitim sonunda Doğrulama kümesi üzerinde “0.1035” kayıp fonksiyon değeri ve “0.9778” doğruluk değeri elde etmiş bulunmaktayız. Şekil 4.6’daki parametre değerleri aynı şekilde oluşturulan özelleştirilmiş model için de geçerli olurken, giriş matrisleri 224,224 ve çıkış değeri 3 olarak değiştirilmiştir.

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	$7 \times 7, 64, \text{stride } 2$
conv2_x	$56 \times 56 \times 64$	$3 \times 3 \text{ max pool, stride } 2$ $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	$7 \times 7 \text{ average pool}$
fully connected	1000	$512 \times 1000 \text{ fully connections}$
softmax	1000	

Şekil 4.6. Resnet18 mimarisi – şekil özelleştirilecek

BÖLÜM 5. SONUÇLAR VE ÖNERİLER

Elde edilen veri setlerini modele sokularak yaklaşık %94 oranında bir başarımla elde edilmektedir. Buradaki doğruluk oranını test ve eğitim veri setlerindeki görüntü sayılarının düşük olmasından dolayı yüksek çıkmaktadır. Modelin doğru seçimi ve veri seti yapısının da modele uygunluğu da bu oranın etkili bir oranda artmasına sebep olmaktadır. Test veri setinin artırılması durumunda doğruluk oranının da azalması mümkündür. Ancak yeterli veri setinin elde edilmesi durumunda model doğruluk oranının azalması durumunda bunun yeniden artırılması için birkaç farklı işlemler yapılabilmektedir. Bunlardan başlıcaları şunlardır:

- Veri setlerinin artırılması ile birlikte bu veri setinin belirli bir oranda ayarlanması gerekebilir. Bölüm 2’de bahsedildiği gibi genellikle %80 model öğrenme veri seti ayarlanırken %20’lik bir test veri seti de eklenebilir. Ancak çok fazla verinin model tarafından öğrenmesi de modelin ezberleme (overfitting) ihtimali de ortaya çıkarmaktadır ve bu da riskli bir durumdur[10][8].
- Veri setinin değiştirilmesi:
 - Farklı veri seti kullanılan model üzerinde daha yüksek başarımla çalışabilir[3].
- Test veri setindeki görüntülerin netliklerinin değiştirilme:
 - Eğer görüntü model tarafından algılanamıyorsa yani öğrendiği özellik haritasındaki sınıflandırma özellikleri tam olarak eşleşmiyorsa modelin yanlış tahminlemesine neden olabilmektedir[7].
- Hiper parametrelerin değerlerinin değiştirilmesi (epoch (yaklaşım) sayısının artırılması, tresholding değerlerinin azaltılması/artırılması vb.),

- Modele ilave katmanlarının eklenmesi veya farklı bir modelin seçilmesi:
 - Eğer modelin özellik haritasını çıkarmasında yardımcı olabilecekse denenebilir, ancak modelin doğruluk oranını düşürebilme ihtimali göz önünde bulundurulmalıdır[8].
 - Ezberlemeyi önlemek için farklı oranlarda dropout katmanlarının eklenmesi/ Dropout katman değerlerinin değiştirilmesi[6].
 - ResNet18 modeli yerine veri setine daha uygun modeller denenebilir. Böylelikle model karşılaştırılması da yapılarak hangisinin daha verimli olabileceği de gözlemlenebilmektedir. Şekil 4.7’de [10] numaralı makale üzerinde x-ray yani röntgen görüntüleri farklı modeller ile denenerek farklı doğruluk oranları elde edilmiştir. Makaledeki veri setini en yüksek oranda DenseNet121 sinir ağı modeli öğrenmiş olduğunu göstermektedir.

Yukarıda açıklanan maddelere göre model doğruluk oranları artabilmeye veya azalabilme ihtimallerinin yanı sıra, ezberleme vb. gibi durumlarla da karşı karşıya kalınabilir.

KAYNAKLAR

- [1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. URL <http://www.deeplearningbook.org>.
- [2] Mustafa Ghaderzadeh, Farkhondeh Asadi, Deep Learning in the Detection and Diagnosis of COVID-19 Using Radiology Modalities: A Systematic Review, Mayıs, 2021.
- [3] S. Wang, B. Kang, A deep learning algorithm using CT images to screen for coronavirus disease (COVID19), 24 Nisan 2020.
- [4] Deep Learning applications for COVID-19, C. Shorten, M. Khoshgoftaar, B.Furht, 11 Ocak 2021.
- [5] Resul Bütüner, M. Hanefi Calp, Derin Öğrenme ve Makine Öğrenmesi Yöntemleri Kullanılarak Akciğer Tomografi Görüntülerinden COVID-19 Tespiti, 27 Ağustos 2020.
- [6] İ. Ayaz, F.Kutlu, V.Tümen, Derin Öğrenme Modellerini Kullanarak X-Ray Görüntülerinden COVID-19'lu Hastaların Aşırı Öğrenme Makineleri ile Tespiti, 4-6 Aralık 2020.
- [7] Yazeed Zoabi, Shira Deri-Rozov, Noam Shomron, Machine learning-based prediction of COVID-19 diagnosis based on symptoms, 4 Ocak 2021.
- [8] Mesut Toğaçar, Burhan Ergen, Zafer Cömert, COVID-19 detection using deep learning models to exploit Social Mimic Optimization and structured chest X-ray images using fuzzy color and stacking approaches, Haziran 2020.
- [9] Doç.Dr.Erkin Dinçmen, Işık Üniversitesi, Makine Öğrenmesi ve Covid-19, 2021.
- [10] Bozkurt Ferhat, Derin Öğrenme Tekniklerini Kullanarak Akciğer X-Ray Görüntülerinden COVID-19 Tespiti, 15 Nisan 2021.
- [11] Murat Semerci, Barış Yaşın, Chapter 7, Convolutional Modern/ResNet, Dive into Deep Learning, 2016.
- [12] Kumar S., Mishra S, Deep Transfer Learning-Based COVID-19 prediction using Chest X-rays medRxiv, 14 Mayıs 2020.
- [13] WHO, Coronavirus disease (COVID-19) Pandemic.

ÖZGEÇMİŞ

Şükrü Ufuk Aksoy, 08.08.1994 tarihinde İstanbul'da doğdu. İlk ve ortaöğretimini İstanbul'da tamamladı. 2012 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği bölümüne girdi. Kartal Belediyesi Bilgi İşlem Daire Başkanlığı'nda(1 ay) donanım, Evyap A.Ş Bilgi Sistemleri ve Süreç Yönetimi departmanında(1 ay) yazılım stajını tamamladı. Aktif olarak web geliştirme ile ilgilenmektedir. Sakarya Üniversitesi Bilgisayar ve Bilişim Bilimleri Fakültesi Bilgisayar Mühendisliği'nde 4. sınıf öğrencisidir.

Gökhan SIBIÇ, 13.10.1995 tarihinde İstanbul'da doğdu. İlk ve orta öğretimini İstanbul'da tamamladı. 2013 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği bölümünü kazandı. Bir yandan da ilk olarak Medipol Üniversitesi Bilgi Teknolojileri Dairesi'nde Donanım stajını, son olarak da Analiz Teknoloji Merkezi'nde Yazılım stajını tamamladı. Aktif olarak İstanbul'da iş hayatına devam etmektedir. Sakarya Üniversitesi Bilgisayar Mühendisliği bölümüne de 4. sınıfta devam etmektedir.

EMRE ÇAKMAK 16.03.1998 tarihinde kocaeli'de doğdu.ilk ve ortaöğretimini Kocaeli'de tamamladı.liseyi Gölcük anadolu öğretmen lisesinde tamamladı.2016 yılında sakarya üniversitesi bilgisayar mühendisliğini kazandı.1 yıl ingilizce hazırlık eğitimininden sonra 2017 yılında bilgisayar mühendisliği bölümüne başladı.1 ay uyumsoft bilgi teknolojileri şirketinde stajını tamamladı.web geliştirme,mobil geliştirme üzerine çalışmalar yapmaktadır.sakarya üniversitesi bilgisayar mühendisliği 4. sınıfta devam etmektedir.

BSM 498 BİTİRME ÇALIŞMASI DEĞERLENDİRME VE SÖZLÜ SINAV TUTANAĞI

KONU : KORONAVİRÜSÜN DERİN ÖĞRENME TEKNİKLERİ İLE TESPİT EDİLMESİ

ÖĞRENCİLER (Öğrenci No/AD/SOYAD):

G121210053 – ŞÜKRÜ UFUK AKSOY

B130910044 - GÖKHAN SIBIÇ

B161210057 - EMRE ÇAKMAK

Değerlendirme Konusu	İstenenler	Not Aralığı	Not
Yazılı Çalışma			
Çalışma klavuza uygun olarak hazırlanmış mı?	x	0-5	
Teknik Yönden			
Problemin tanımı yapılmış mı?	x	0-5	
Geliştirilecek yazılımın/donanımın mimarisini içeren blok şeması (yazılımlar için veri akış şeması (dfd) da olabilir) çizilerek açıklanmış mı?			
Blok şemadaki birimler arasındaki bilgi akışına ait model/gösterim var mı?			
Yazılımın gereksinim listesi oluşturulmuş mu?			
Kullanılan/kullanılması düşünülen araçlar/teknolojiler anlatılmış mı?			
Donanımların programlanması/konfigürasyonu için yazılım gereksinimleri belirtilmiş mi?			
UML ile modelleme yapılmış mı?			
Veritabanları kullanılmış ise kavramsal model çıkarılmış mı? (Varlık ilişki modeli, noSQL kavramsal modelleri v.b.)			
Projeye yönelik iş-zaman çizelgesi çıkarılarak maliyet analizi yapılmış mı?			
Donanım bileşenlerinin maliyet analizi (prototip-adetli seri üretim vb.) çıkarılmış mı?			
Donanım için gerekli enerji analizi (minimum-uyku-aktif-maksimum) yapılmış mı?			
Grup çalışmalarında grup üyelerinin görev tanımları verilmiş mi (iş-zaman çizelgesinde belirtilebilir)?			
Sürüm denetim sistemi (Version Control System; Git, Subversion v.s.) kullanılmış mı?			
Sistemin genel testi için uygulanan metotlar ve iyileştirme süreçlerinin dökümü verilmiş mi?			
Yazılımın sızma testi yapılmış mı?			
Performans testi yapılmış mı?			
Tasarımın uygulamasında ortaya çıkan uyumsuzluklar ve aksaklıklar belirtilerek çözüm yöntemleri tartışılmış mı?			
Yapılan işlerin zorluk derecesi?	x	0-25	
Sözlü Sınav			
Yapılan sunum başarılı mı?	x	0-5	
Soruları yanıtlama yetkinliği?	x	0-20	
Devam Durumu			
Öğrenci dönem içerisindeki raporlarını düzenli olarak hazırladı mı?	x	0-5	
Diğer Maddeler			
Toplam			

DANIŞMAN (JÜRİ ADINA):

DANIŞMAN İMZASI: